

IOANA BĂRBAT

MIHAI RADU DUMITRESCU

STELIAN NICULESCU

CAZIMIR MACARIE

ANASTASE PITIȘ

# PROGRAMAREA ȘI EXPLOATAREA SISTEMELOR DE CALCUL

Manual pentru licee de matematică-fizică  
cu profil de informatică, clasa a XI-a



IOANA BĂRBAT  
matematician

MIHAI RADU DUMITRESCU  
matematician

STELIAN NICULESCU  
matematician

CAZIMIR MACARIE  
economist

ANASTASE PITIȘ  
matematician

# PROGRAMAREA ȘI EXPLOATAREA SISTEMELOR DE CALCUL

Manual pentru licee de matematică-fizică,  
cu profil de informatică, clasa a XI-a



EDITURA DIDACTICĂ ȘI PEDAGOGICĂ, BUCUREȘTI — 1982



## PARTEA ÎNTÎI

# PROGRAMAREA ÎN LIMBAJUL FORTRAN

### CAPITOLUL I

## NOȚIUNI INTRODUCTIVE

### 1. CONSIDERAȚII GENERALE

Automatizarea unor activități umane a fost avută în vedere din cele mai vechi timpuri, atât sub aspect filozofic și social cît și sub aspect tehnic. Dorința omului de a se elibera de efortul fizic, în folosul activității creatoare, s-a aliniat la efortul general pentru progres. Deceniul al cincilea al secolului al XX-lea a marcat o cotitură de seamă, problema automatizării căpătînd noi valențe, noi dimensiuni, punîndu-se problema automatizării în munca intelectuală. Calculatoarele electronice sînt cele care au marcat acest moment de răscruce.

Încercările de început pe calea automatizării unor activități umane își au originea în secolele trecute. Personalități marcante, cum sînt Napier (1617), Pascal (1642), Babbage (1820), își au numele legate de realizarea unor dispozitive (mașini) de calculat. De fapt, ideea lui Babbage, lansată și susținută între anii 1820—1856, a stat la baza realizării primelor calculatoare electronice. Aparatul imaginat de el era prevăzut să efectueze operații aritmetice, să memoreze atât instrucțiuni (comenzi) cît și rezultate pe care apoi să le furnizeze.

Ca prim descendent al mașinii imaginată de Babbage a fost calculatorul ASCC (Automatic Sequence Controlled Calculator), realizat de cunoscuta firmă IBM în colaborare cu Universitatea Harvard (1944), calculator a cărui greutate era de circa cinci tone și care avea un volum considerabil.

Lui ASCC i-a urmat ENIAC (Electronic Numerical Integrator And Calculator) considerat, în fapt, a fi primul calculator electronic. Acesta din urmă avea în componență circa 20.000 de tuburi electronice și efectua aproximativ 5.000 de adunări pe secundă.

După cum este cunoscut, în funcție de componentele electronice constitutive, calculatoarele electronice sînt împărțite pe generații, evoluția atât din punct de vedere hardware cît și din punct de vedere software fiind spectaculoasă. În zilele noastre microcalculatoarele și minicalculatoarele au căpătat o răspîndire mare, informatica distribuită fiind o realitate.



## 2. LIMBAJE DE PROGRAMARE

Limbajele de programare, ca și limbajele naturale, au la bază:

- un alfabet care este o mulțime finită și nevidă de caractere (litere, cifre, semne speciale);
- un vocabular care este o mulțime de cuvinte, constituită din caractere sau succesiuni finite de caractere;
- o gramatică care este ansamblul unor reguli destinate validării textelor (succesiuni finite de cuvinte), în sensul de a preciza că aparțin sau nu unui limbaj.

Evoluția rapidă în ceea ce privește echipamentele de calcul a impus, în mod firesc, o evoluție remarcabilă a limbajelor de programare. Există astăzi sute de limbaje, ceea ce, în mod natural, impune o clasificare a lor. Dintre multiplele moduri de clasificare se prezintă acela conform căruia există: limbaje de niveluri 0, 1, 2, 3, limbaje specializate, limbaje de tip pseudocod.

Limbajele de nivel 0 sînt limbajele-mașină, fiecare calculator electronic, prin construcție, avînd limbajul său — setul său de instrucțiuni. Pe măsură ce calculatoarele au devenit mai complexe, programarea directă, în limbaj-mașină, a devenit mai greoaie, cu multă muncă de rutină, ceea ce a condus la ideea automatizării ei. Au apărut astfel noi limbaje: limbajele de asamblare.

Limbajele de asamblare, considerate ca fiind de nivel 1, au structură similară cu limbajele-mașină, numai că locul codurilor numerice a fost luat de prescurtări sugestive (mnemonice), iar adresele nu mai sînt absolute (numere) ci sînt simbolice. Trecerea de la un program scris într-un limbaj de asamblare (numit *program* origine sau *sursă*) la unul în limbaj-mașină se face pe baza unui program traducător corespunzător, numit *asamblor*. Programul astfel obținut este numit program rezultat sau *program obiect*.

Limbajele de nivel 2 sînt numite macro-asambloare, fiind tot limbaje simbolice. Au în plus, față de cele de nivel 1, macro-instrucțiuni. În felul acesta programele se scriu mai condensat, existînd mai puține surse de erori.

Nivelul 3 este constituit din limbajele evolute. Cercetările din perioada 1952—1956 au dus la crearea acestei clase de limbaje, evenimentul fiind marcat de apariția limbajului FORTRAN (FORmula TRANslator). Prin structură sa, este destinat rezolvării problemelor tehnico-științifice (date puține cu calcule complexe). Fiind însă un limbaj universal, poate fi utilizat și în rezolvarea altor tipuri de probleme. Programul traductor se numește *compilator* FORTRAN și realizează traducerea din limbajul FORTRAN în limbajul-mașină. Primele specificații ale limbajului au apărut în 1954. Cîteva dintre variantele sale s-au bucurat de stabilitate și au avut o circulație mai mare: FORTRAN II (1958), Basic FORTRAN, FORTRAN IV (1962) și FORTRAN standard (1966) numit, în cele ce urmează, FORTRAN. În anul 1976 s-a elaborat un proiect de standardizare, ANS FORTRAN (ANS prescurtare de la American National Standards Committee), prezentat în două variante: limbajul complet și o restricție (ca revizie a variantei FORTRAN din 1966).

Deceniul 1960—1970 a constituit o perioadă de maturizare în domeniul limbajelor de programare.

Limbajul FORTRAN a fost primul limbaj de programare standardizat, standardizarea oferind o serie de avantaje (ușurință în utilizare pe scară largă, compatibilitate cu variantele anterioare, facilități de extensibilitate).

În 1960 a apărut un nou limbaj, COBOL (COmmon Business Oriented Language), destinat rezolvării problemelor economice — volum mare de informații, cu calcule simple (primele specificații COBOL s-au făcut în 1959).



El este superior limbajului FORTRAN în ceea ce privește lucrul cu fișiere. Și limbajul COBOL este un limbaj de tip universal, ceea ce permite folosirea lui în rezolvarea și a altor tipuri de probleme (de traducere automată, de calcule științifice etc.).

Tot în anul 1960 a apărut limbajul ALGOL (ALGOritmic Language), limbaj universal care a preluat ce au avut mai bun limbajele ce l-au precedat. Are o structură foarte bine pusă la punct. Fiind complex, este mai puțin răspândit în rândul utilizatorilor, dar este folosit ca limbaj de publicare a programelor (există variantele ALGOL-60, ALGOL-68).

Limbajul PL/1 (Programming Language One), sinteză a limbajelor FORTRAN, COBOL, ALGOL a apărut în anul 1964.

Din rândul limbajelor specializate cităm: limbajele conversaționale (BASIC, APL etc.); limbajele pentru probleme de simulare (GPSS, SIMULA etc.).

### 3. STRUCTURA PROGRAMELOR FORTRAN

Instrucțiunile limbajului FORTRAN sînt de două feluri (v. tabelul I.1):

— instrucțiuni neexecutabile, cele care furnizează informații necesare procesului de prelucrare;

— instrucțiuni executabile, cele din care se generează efectiv instrucțiuni ale programului obiect.

TABELUL I.1

Nr. crt.	Instrucțiuni	
	neexecutabile	executabile
1	BLOCK DATA	ASSIGN
2	CHARACTER *	BACKSPACE
3	COMMON	CALL
4	COMPLEX	CLOSE *
5	DATA	CONTINUE
6	DIMENSION	DECODE
7	DOUBLE PRECISION	DO
8	ENTRY	ENCODE
9	EQUIVALENCE	END FILE
10	EXTERNAL	FIND
11	FORMAT	GO TO necondiționat
12	FUNCTION	GO TO impus
13	IMPLICIT	GO TO calculat
14	INTEGER	IF aritmetic
15	INTRINSIC *	IF logic
16	Instrucțiuni de definire a unor funcții	INQUIRE *
17	LOGICAL	Instrucțiuni de atribuire
18	NAMelist	OPEN *
19	PARAMETER *	PAUSE
20	PROGRAM *	PRINT
21	REAL	PUNCH
22	SAVE *	READ
23	SUBROUTINE	RETURN
		REWIND
		STOP
		WRITE



Instrucțiunile marcate cu \* nu sînt acceptate de varianta actuală a compilatorului FORTRAN FELIX C-256.

În general, structura unei instrucțiuni este următoarea (sînt exceptate comentariile și instrucțiunile specifice sistemului de operare):

*etichetă c instrucțiunea propriu-zisă*

Eticheta este un număr natural cuprins între 1 și 99999, iar *c* este un caracter care dacă este diferit de spațiu sau zero semnifică o continuare (numai în zona instrucțiune propriu-zisă) a unei instrucțiuni care nu a încăput pe rîndul anterior.

#### Observație

Toate instrucțiunile neexecutabile sînt neetichetabile (afară de instrucțiunea FORMAT), iar toate cele executabile sînt etichetabile.

Programele scrise în limbajul FORTRAN au o structură similară cu cea dată în figura I.1.

Ceea ce rezultă imediat din figura prezentată este faptul că nivelele 5 și 6 sînt obligatorii, existînd cel puțin o instrucțiune executabilă și sfîrșitul fizic (END) al programului. De asemenea, instrucțiunile FORMAT pot să apară oriunde la nivelele 2, 3, 4, 5, iar comentariile pot să figureze la oricare din cele 6 nivele. Un rînd comentariu se marchează prin prezența caracterului C în coloana 1 și în coloanele următoare comentariul respectiv.

Limbajul FORTRAN FELIX C-256 nu acceptă instrucțiunea PROGRAM *a* unde *a* este numele simbolic al programului principal. În FORTRAN STANDARD și FORTRAN ANS această instrucțiune este acceptată. În schimb, FORTRAN STANDARD nu acceptă proceduri de tip BLOCK DATA.

Așadar, programele principale în cazul lucrului la FELIX C-256, nu au instrucțiuni de nivel 1, orice job FORTRAN admițînd un singur program principal.

NIVEL	TIP DE INSTRUCȚIUNI	INSTRUCȚIUNEA COMENTARIU	INSTRUCȚIUNEA FORMAT	OBSERVAȚII
1	TIPUL PROGRAMULUI PROGRAM FUNCTION SUBROUTINE BLOCK DATA			ALTE NIVELE POATE FI
2	DECLARAȚII			POT FI DE TIP DECLARAȚIE IMPLICIT PRIN NUME
3	INSTRUCȚIUNEA DATA			SE POATE SA APARA LA NIVELUL 2, 3, 4, 5
4	FUNCȚIA FORMULA			POT FI DE TIP
5	INSTRUCȚIUNI EXECUTABILE			
6	INSTRUCȚIUNEA END			

Fig. I. 1.



#### 4. FORMULARUL DE PROGRAMARE

Limbajul FORTRAN nu figurează nici în rîndul limbajelor cu scriere impusă strict, nici în cadrul limbajelor cu scriere liberă.

Ceea ce este strict necesar să se respecte se rezumă la:

- scrierea etichetelor instrucțiunilor în cîmpul 1—5 al formularului de programare (al cartei), alinierea în cadrul acestui cîmp nefiind obligatorie (se preferă alinierea la dreapta);

- coloana șase este destinată eventualelor continuări de scriere de pe un rînd pe altul (în cazul instrucțiunilor prea lungi); cînd coloana șase conține blank sau zero nu este vorba de continuare;

- scrierea instrucțiunilor se face în porțiunea 7—72 a formularului de programare, conform structurii ce o au (nu se pot scrie mai multe instrucțiuni pe un rînd).

Referitor la aspectul de libertate de scriere se citează faptul că în cîmpul 7—72 este liberă utilizarea spațiului, ceea ce oferă posibilitatea scrierii unor programe care să aibă calitatea de a fi mai ușor înțelese (lizibilitate sporită).

Cîmpul 73—80 este la libera alegere a programatorului, nefiind interpretat de compilator.

##### Observație

În majoritatea cazurilor, instrucțiunile FORTRAN sînt aliniate la stînga în cîmpul 7—72.



## CAPITOLUL II

# ELEMENTE DE BAZĂ ALE LIMBAJULUI FORTRAN

### 1. SETUL DE CARACTERE

Există trei tipuri de caractere și anume:

- literele majuscule A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z;
- cifrele sistemului de numerație cu baza 10 (0, 1, 2, 3, 4, 5, 6, 7, 8, 9);
- caracterele speciale (+ - \* / =  $\square$  . ( ) , & ' ).

Cu  $\square$  s-a notat spațiul (blancul). Când se utilizează nu se va scrie  $\square$  ci se va lăsa un spațiu.

#### Observație

Limbajul FORTRAN ANS mai cuprinde în rândul caracterelor speciale încă două și anume: simbolul monetar (\$) și două puncte (:).

### 2. CONSTANTE

Există trei feluri de constante: constante logice, constante numerice și constante alfanumerice.

Înainte de a trece la prezentarea lor, menționăm că limbajul FORTRAN admite șase tipuri de date și anume:

- date de tip logic;
- date de tip numeric întreg;
- date de tip numeric real;
- date de tip numeric complex;
- date de tip dublă precizie;
- date de tip caracter (alfanumerice).

a. Constantele logice sînt:

.TRUE.  
.FALSE.

și pun în evidență *valoarea logică adevărat* (.TRUE.) și *vălcarea logică fals* (.FALSE.).

În ceea ce privește reprezentarea datelor de tip logic în memoria centrală, precizăm că *valoarea logică adevăr* se pune în evidență prin faptul că zona de memorie în care se reprezintă este completată în binar numai cu unu iar la *valoarea logică fals*, zona respectivă se completează numai cu zero. În mod obișnuit reprezentarea se face pe cuvinte de memorie.

b. Constantele numerice sînt de patru tipuri: întregi, reale, complexe, dublă precizie.

Constantele de tip *întreg* sînt numere întregi în sensul cunoscut. Reprezentarea lor în memoria centrală a calculatorului se face exact (fără rotunjiri) prin trecerea din baza 10 în baza 2. De obicei, reprezentarea se face pe un



cuvînt de memorie, în cod complementar, ceea ce face ca să se poată lucra cu numere întregi care sînt în intervalul  $[-2^{31}, 2^{31} - 1]$ .

*Constantele de tip real* sînt numere în sensul cunoscut numai că în loc de virgulă se utilizează punctul pentru a separa partea întreagă de partea fracționară. Se poate să lipsească sau partea întreagă sau partea fracționară (de exemplu, .5 în loc de 0.5, sau 1. în loc de 1.0).

Pe lîngă acest mod cunoscut de a defini constante de tip real, limbajul FORTRAN mai acceptă și alte forme de constante de tip real și anume:

$$\begin{aligned} x & E \ n \\ x & E \ + \ n \\ x & E \ - \ n \\ x & + \ n \\ x & - \ n \end{aligned}$$

unde  $x$  este constantă de tip real în sensul cunoscut (cu punctul ca marcă zecimală), iar  $n$  este număr natural. Ultimele două tipuri de constante sînt valabile numai la citiri de date de pe suporți (de obicei, de pe cartele).

#### Exemple

-0.0125	149.0	+.1	4953.E-4
-0.	-1933.	-05.5	4953.-4
-0.00	-.0	-1.	-3.E-42
0.4E2	5.E-1	-1.5E03	1.-48
0.4+2	5.-1	-1.5+03	0.5+10

*Constantele de tip complex* sînt definite ca perechi de constante de tip real. Prima componentă a perechii este partea reală a numărului complex, iar a doua este partea imaginară a numărului complex.

#### Exemple

	Numărul complex
(5.25,.5)	$5.25 + 0.5i$
(0.,125.)	$125i$
(1.,1.)	$1+i$

#### Observație

Este obligatoriu ca cele două elemente ale perechii să fie constante de tip real; nu este acceptată drept constantă complexă, de exemplu, perechea

$$(1,4)$$

deoarece cele două constante 1 și 4 sînt întregi, nu reale.

În ceea ce privește reprezentarea, se face pe un cuvînt dublu; în prima jumătate se reprezintă (în virgulă mobilă) partea reală, iar în a doua jumătate se reprezintă partea imaginară.

*Constantele de tip dublă precizie* sînt definite ca și constantele de tip real varianta cu E, locul lui E fiind luat de D.

#### Exemple

0.05D-4	-.054D14
1.0D+1	-.999D7

Reprezentarea lor în memoria centrală se face pe cuvinte duble, în convenția de virgulă mobilă, al doilea cuvînt fiind rezervat pentru încă 8 cifre



ale mantisei. Deci, primul octet din cei 8 este rezervat caracteristicii, iar ceilalți 7 sînt destinați reținerii a 14 cifre hexazecimale ale mantisei.

2. Constantele de tip caracter (alfanumerice) sînt șiruri de caractere. În limbajul FORTRAN se definesc în două moduri:

*n*H șir de *n* caractere

sau

'șir de caractere'

Datorită utilizării apostrofului în definirea constantelor (a doua definiție) există o restricție în ceea ce privește utilizarea apostrofului în cadrul șirului de caractere. Prezența a 2K semne apostrof echivalează cu luarea în considerație, în șir, a K semne apostrof. Reprezentarea în memorie se face caracter pe octet.

*Exemple*

4HABCD sau 'ABCD' 1HH sau 'H' 1H' sau ''''  
3HX'Y sau 'X''Y' 2H+\* sau '+\*' 2H'' sau ''''''

### 3. IDENTIFICATORI ȘI ETICHETE

Identificatorii (numele simbolice) sînt succesiuni de litere sau cifre, primul dintre caractere fiind obligatoriu literă.

În privința lungimii (numărul de caractere ce compun un identificator), majoritatea compilatoarelor se limitează la 6 caractere maximum. La FORTRAN FELIX C-256 această limitare nu se pune. În caz că sînt mai multe de 8 caractere se consideră semnificative primele 8.

*Exemple*

AUX	T	TAU
INDICE	EC	ZETA

Etichetele sînt numere naturale din intervalul [1,99999].

### 4. VARIABLE

Variabilele, ca și constantele, sînt de cele șase tipuri. După cum se va vedea, aceste tipuri vor rezulta implicit (din nume) sau explicit (prin declarații de tip).

Există două categorii de variabile: variabile simple (neindexate), desemnate printr-un nume simbolic (identificator), variabile indexate (vectori, matrice) desemnate printr-un nume simbolic, referirea la componente făcîndu-se prin precizarea indicilor sau expresiilor indiciale, între paranteze, după numele variabilei (indicii sînt separați prin virgulă).

Pentru variabilele de tip numeric întreg și real, dacă numele începe cu una din literele I, J, K, L, M, N, înseamnă că este vorba de o variabilă de tip întreg, iar în caz contrar este vorba de variabilă de tip real. Explicit, tipul poate fi precizat prin declarații INTEGER, respectiv REAL. Dacă se dorește să se impună tipul tot prin prima literă a numelui, se utilizează declarația IMPLICIT.



Tipurile complex, dublă precizie, logic, implicit rezultă numai explicit, prin declarații (COMPLEX, DOUBLE PRECISION, LOGICAL, IMPLICIT). Singurul tip care rezultă din contextul utilizării este tipul alfanumeric.

#### Observație

La FORTRAN FELIX C-256 este posibil ca tuturor variabilelor să li se rezerve cuvinte duble dacă se pune parametrul DBL în cartela COMPILE. Acest lucru este util când se dorește să se lucreze cu tipul complex în dublă precizie (evident, există o risipă de memorie, pentru eventualele variabile de tip întreg și logic, spațiul dublu nefiind folositor).

Limbajul BASIC FORTRAN și FORTRAN STANDARD admit ca indici numai expresii de forma:

$$\begin{aligned} & i \\ & v \\ & v \pm a \\ & a * v \\ & a * v \pm b \end{aligned}$$

unde  $a$ ,  $b$ , sînt constante întregi, iar  $v$  este variabilă de tip întreg. FORTRAN FELIX C-256 admite ca indice orice expresie numerică, luînd drept indice partea întreagă a rezultatului evaluării (nu admite ca indice o variabilă de tip real).

#### Exemple

X11	T(1,2,I,J)
A(3,4)	WWW(K,J-K,3)
AB(2*I,J+3)	ADUNA(I,1,1)

## 5. LISTE DE VARIABILE

O listă de variabile este constituită din variabile simple, componente de variabile indexate sau nume de variabile indexate (în acest caz numele arată că sînt luate în considerație, în listă, toate componentele ei, în ordinea din memorie), separate prin virgulă.

#### Exemple

A,B,C,D  
 A(1), A(2), AUX, K  
 V,W(I), TAU(2\*I, K-1), Z(1), Z(3), Z(5), Z(7)  
 X(2), X(4), X(6), X(8), X(10), X(12)  
 A(1), B(1), A(3), B(3), A(5), B(5), A(7), B(7), A(9), B(9)  
 A(1,1), A(1,2), A(1,3), A(2,1), A(2,2), A(2,3)  
 Y1(1), Y2(2), Y1(2), Y2(4), Y1(4), Y1(3), Y2(6), Y1(4), Y2(8)

Ultimele patru exemple de liste permit scriere prescurtată:

(X(I), I = 2, 12, 2)

↓  
 rația (cînd rația lipsește se ia automat 1)  
 ↓  
 valoarea finală a lui I  
 ↓  
 valoarea de început a lui I



$(A(K), B(K), K = 1, 9, 2)$   
 $(A(I, J), J = 1, 3, I = 1, 2)$   
 $(Y1(I), Y2(2, I), I = 1, 4)$

## 6. EXPRESII

Limbajul FORTRAN acceptă trei tipuri de expresii: numerice (aritmice), relaționale, logice.

a. **Expresiile numerice** sînt reguli de calcul numeric, operatorii fiind:

$+, -, *, /, **$

respectiv pentru adunare, scădere, înmulțire, împărțire, ridicare la putere.

Ca operanzi se pot lua:

- constante numerice (fără tipul  $n \pm e$ );
- variabile numerice;
- funcții numerice.

Prioritățile operațiilor sînt:

- ridicări la putere ( $**$ );
- înmulțiri ( $*$ ), împărțiri ( $/$ );
- adunări ( $+$ ), scăderi ( $-$ ).

La priorități egale (operații succesive de aceeași prioritate), se aplică regula dreapta-stînga la ridicările la putere și stînga-dreapta la celelalte.

*Exemple*

Expresia	Cum se evaluează
$A+B+C$	$(A+B)+C$
$A*B/C$	$(A*B)/C$
$A**B**C$	$A^{(B^C)}$
$SIN(X)+COS(X)$	$SIN(X)+COS(X)$

Prin utilizarea parantezelor se poate impune o anumită ordine de efectuare a calculelor. În cazul expresiilor cu paranteze, evaluarea se face cu prioritățile menționate, începînd cu parantezele cele mai interioare.

Expresiile sînt omogene dacă toți operanzii sînt de același tip și sînt neomogene în caz contrar. În ultimul caz, în timpul evaluării expresiilor se fac conversii, orice operație fiind precedată de conversia (dacă operanzii nu au același tip) operandului cu tipul mai slab în tipul mai tare. Tipurile, în ordinea crescîndă a tăriei, sînt:

- întreg (I);
- real (R);
- dublă precizie (DP);
- complex (C).

Evaluările pentru operațiile  $+, -, *, /$  se fac în conformitate cu tabelul II.1.

TABELUL II.1

$+, -, *, /$	I	R	DP	C
I	I	R	DP	C
R	R	R	DP	C
DP	DP	DP	DP	C
C	C	C	C	C



Pentru ridicarea la putere, evaluările se fac conform tabelului II.2.

TABELUL II.2

..	I	R	DP	C
I	I	R	DP	—
R	R	R	DP	—
DP	DP	DP	DP	—
C	C	—	—	—

S-au marcat cu liniuță cazurile nepermise.

b. **Expresiile relaționale.** Operatorii relaționali permisi în FORTRAN sînt prezentați în tabelul II.3.

TABELUL II.3.

Nr. crt.	Operatorul relațional	Codificarea în FORTRAN	Denumirea engleză
1	<	.LT.	Less Than
2	≤	.LE.	Less or Equal
3	=	.EQ.	Equal
4	≥	.GE.	Greater or Equal
5	>	.GT.	Greater Than
6	≠	.NE.	Not Equal

Două expresii numerice care nu au operanzi de tip complex, legate prin unul din operatorii relaționali, constituie o expresie relațională.

Evaluarea unei expresii relaționale se face astfel:

- se evaluează cele două expresii numerice;
- se aduc cele două valori rezultate la același tip (dacă este cazul) și anume la tipul mai tare;
- dacă relația respectivă are loc atunci i se asociază expresiei relaționale valoarea logică adevăr, iar în caz contrar valoarea logică fals.

*Exemple*

(A\*B/C-I).GT.0  
A\*\*2+B\*\*2.GE.XA+Y  
SIN(X).GE.0

c. **Expresiile logice.** Operatorii logici sînt:

.NOT.  
.AND.  
.OR.

care corespund, respectiv, funcțiilor logice: negație, conjuncție, disjuncție. Prioritățile lor sînt în ordinea dată. Ca și la expresiile numerice, utilizarea parantezelor facilitează impunerea unei anumite ordini de efectuare a operațiilor logice.



În componența unei expresii logice pot intra:

- constante logice;
- variabile logice;
- funcții logice;
- expresii relaționale.

*Exemple*

A.AND.B  
A.GT.0.AND.NOT.B  
X.OR.Y.AND.NOT.(Z.OR.T)  
(A+B).GT.0.AND.(C.NE.0)

### ÎNTREBĂRI ȘI PROBLEME

1. Câte tipuri de date acceptă limbajul FORTRAN și care sînt acestea?
2. Care sînt formele de constante de tip real?
3. Care dintre formele de constante reale nu pot fi utilizate ca operanzi din cauza ambiguităților ce le-ar produce?
4. Să se reprezinte numărul  $x = -0.625$  în convenția virgulă mobilă, precizie simplă.
5. Care este reprezentarea celui mai mare număr întreg pe un cuvînt de memorie?
6. Ce reprezentare are cel mai mic număr întreg (pe un cuvînt de memorie)?
7. În cîte feluri se poate defini, drept constantă FORTRAN, șirul de caractere:

ABCD = : Z" P3437XYZTUW.IKJH

8. Care este lungimea, în general, pentru identificării FORTRAN?
9. Ce lungime pot avea identificatorii acceptați de limbajul FORTRAN FELIX C-256?
10. Limbajul FORTRAN FELIX C-256 acceptă o variabilă de tip real ca indice al unei componente de variabilă indexată?
11. O expresie de tip numeric poate fi utilizată ca indice al unei componente de variabilă indexată?



## CAPITOLUL III

### INSTRUCȚIUNI PENTRU SCRIEREA PROGRAMELOR CU STRUCTURĂ LINIARĂ

#### 1. INSTRUCȚIUNI DE DECLARARE

Aceste instrucțiuni sînt neexecutabile în sensul în care a rezultat în paragraful 3 al capitolului I și sînt neetichetabile.

După cum s-a văzut în capitolul anterior, există șase tipuri de date dintre care numai tipul alfanumeric rezultă din contextul utilizării, restul de cinci tipuri, în cazul că nu se lucrează cu tipurile întreg și real predefinite, sînt sugerate de cuvintele:

INTEGER  
REAL  
DOUBLE PRECISION  
COMPLEX  
LOGICAL

Aceste cuvinte definesc de fapt, instrucțiunile de declarare respective. În cazul că se dorește un tip implicit sugerat de nume, altul decît cel predefinit, se utilizează instrucțiunea

#### IMPLICIT

a cărei structură se va da după prezentarea instrucțiunii DIMENSION și a celor cinci instrucțiuni de declarare.

• Declararea variabilelor indexate cu tipul predefinit prin nume (întregi, reale) se face cu *instrucțiunea* DIMENSION care are structura:

#### DIMENSION listă

în listă scriindu-se, separate prin virgulă, ultimele componente ale variabilelor indexate cu care se lucrează.

În cazul că este vorba de matrice (de doi indici, de exemplu), rezervarea spațiului în memorie se face pe coloane. De exemplu, pentru o matrice A cu două linii și trei coloane se rezervă un spațiu de șase cuvinte distribuite astfel:

$A(1,1)$	$A(2,1)$	$A(1,2)$	$A(2,2)$	$A(1,3)$	$A(2,3)$
↓	↓	↓	↓	↓	↓
coloana 1		coloana 2		coloana 3	

#### Observație

Ca regulă generală, cînd este vorba de mai mulți indici, viteza de variație (automată) a indicilor este de la cel mai din stînga la cel mai din dreapta.

Instrucțiunile de declarare au o structură similară și anume:

#### tip listă

unde tip este tipul și poate fi una din cele cinci posibilități (INTEGER, REAL, DOUBLE PRECISION, COMPLEX, LOGICAL), iar listă cuprinde



variabilele care urmează ca, în programul în care apare o astfel de declarație, să fie de tipul specificat. Dacă este vorba de variabile indexate atunci în listă se specifică numai ultima componentă, în cazul acesta declararea de tip rezolvînd și problema rezervării de spațiu de memorie (are rol și de DIMENSION).

De fapt, forma generală a instrucțiunilor de declarare este:

tip [*\* lungime*] listă

unde lungimea (în octeți) este opțională (lucru marcat de paranteza dreaptă). Cînd nu se specifică (ideea în care s-a prezentat), se rezervă fiecărei variabile din listă cîte un cuvînt, exceptînd dubla precizie cînd se rezervă cuvinte duble. Varianta actuală a compilatorului la FELIX C—256 nu acceptă altă lungime decît 4 sau 8, caz în care nu trebuie specificată, ea rezultînd din tip.

#### Observație

Tipul explicit este mai puternic decît cel predefinit.

● *Instrucțiunea IMPLICIT* are ca rol să impună, în mod voit, în cadrul programului unde este prezentă, ca variabilele ce încep cu anumite litere să aibă un tip dorit. Structura este:

IMPLICIT tip (*listă de litere*)

unde tip este unul din cele cinci tipuri posibile, după care se specifică între paranteze literele (separate prin virgulă) care se dorea ca fiind litere de început, să definească variabile de tipul specificat.

#### Observații

1. Listele de litere succesive în alfabet se pot prescurta scriînd prima literă — ultima literă din succesiunea respectivă (vezi ultimele exemple dintre cele date mai jos).

2. Punînd virgulă după paranteza închisă a listei de litere, se poate lua un nou tip și o nouă listă ș.a.m.d. (vezi penultimul exemplu de instrucțiune IMPLICIT).

3. De reținut că DIMENSION nu rezolvă și problema tipului ci numai cea a dimensionării, în afara altor specificații de tip însemnînd că sînt avute în vedere tipurile întreg sau real predefinite.

#### Exemple

```
INTEGER A,B,AUX,V(5), AR(10,5)
REAL I(4),K,U(15), E14(4,3,9)
COMPLEX A(5),TRUE(10), REAL(5).
LOGICAL L,A,DA(4), N(4,2), IESTE
DOUBLE PRECISION IF, DO, REAL
IMPLICIT REAL (I,J,K)
IMPLICIT INTEGER (A—Z)
IMPLICIT INTEGER (A—I), LOGICAL (J—R), REAL (S—Z)
IMPLICIT DOUBLE PRECISION (A—E,H,J)
DIMENSION A(6), I(9,2), MATRICE (7,7)
```



## 2. INSTRUCȚIUNILE STOP, PAUSE, CONTINUE

Aceste trei instrucțiuni sînt executabile în sensul precizat, putînd fi și etichetate.

Rolul instrucțiunii STOP este acela de a defini sfîrșitul logic al programului (la programele structurate sfîrșitul logic precede imediat sfîrșitul fizic și este unic).

- Structura *instrucțiunii* STOP este următoarea:

STOP *a*

unde *a* poate fi un număr natural sau o constantă alfanumerică (mesaj pentru programator sau operator). Cazul cel mai frecvent este ca *a* să nu apară, deci forma instrucțiunii să fie:

STOP

- *Instrucțiunea* PAUSE are aceeași structură și mod de funcționare ca și STOP numai că ea pune în evidență un stop relativ, cu posibilitatea de reluare, în continuare, a prelucrării. Este folosită fie în depanarea programelor pentru a valida programul pe porțiuni, fie ca moment de întrerupere necesar unor manevre pentru operare. Structura ei este:

PAUSE *a*

*a* avînd aceeași interpretare ca la STOP.

După cum se va vedea, în ideea programării structurate rolul instrucțiunii CONTINUE crește, fiind mult mai prezentă în programe. În mod obișnuit este etichetată, ca structură constînd numai în scrierea cuvîntului CONTINUE în cîmpul 7—72 al formularului de programare.

*Exemple*

```
STOP  
STOP 3  
PAUSE 'MONTAȚI DISC DE LUCRU ȘI CONTINUAȚI LUCRUL'  
PAUSE 1  
CONTINUE
```

## 3. INSTRUCȚIUNI DE ATRIBUIRE

Acest tip de instrucțiuni este destinat precizării valorilor unor variabile fie în momentul compilării (instrucțiunea DATA), fie prin calcul (numeric, logic). În orice caz, valorile sînt generate nu preluate de pe un suport ca în cazul citirilor.

a. *Instrucțiunile de atribuire de tip numeric.* Rolul instrucțiunilor de acest tip constă în aceea că o valoare care rezultă din calcule se atribuie unei variabile (există limbaje care permit atribuire multiplă, valoarea putînd fi atribuită mai multor variabile — de exemplu, ALGOL).

Structura instrucțiunii este:

*V = E*



unde  $v$  este variabilă numerică simplă sau componentă de variabilă numerică indexată, iar  $e$  este expresie numerică.

Executarea instrucțiunii se face fără conversie dacă  $v$  și  $e$  au același tip ( $v$  preia exact ceea ce a rezultat din evaluarea lui  $e$ ) sau cu convertirea rezultatului evaluării lui  $e$  în tipul lui  $v$  și apoi preluare. În ultimul caz, dacă, de exemplu:

$$I = A + J$$

cu  $J=3$  și  $A=1.3$ , atunci  $I$  devine 4 (se ia partea întreagă a rezultatului 4.3 al evaluării).

#### Exemple

Dacă  $A, B, C$  sînt variabile de tip întreg,  $D, E, F$  variabile de tip real,  $G, H, I$  variabile de tip complex, iar  $J$  vector cu cinci componente de tip real, cu valorile:

$$\begin{array}{lll} A=3 & D=1. & G=(1.,2.5) \\ B=2 & E=3.6 & H=(0.,5.) \end{array}$$

atunci se pot da exemplele prezentate în continuare.

Instrucțiunea	Tipul lui		Valoarea lui	Efectul instrucțiunii
	$v$	$e$	$e$	
$C=A*B+E$	I	R	9.6	$C=9$
$F=G+D+H$	R	C	(2.,7.5)	$F=2$
$C=A+B$	I	I	5	$C=5$
$I=E-D$	C	R	2.6	$I=(2.6,0)$
$J(A)=E$	R	R	3.6	$J(3)=3.6$
$J(2*A-D)=D-E$	R	R	-2.6	$J(5)=-2.6$

b. **Instrucțiunile de atribuire de tip logic** sînt destinate precizării valorilor unor variabile de tip logic. Structura lor este de forma:

$$v = e$$

unde, de data aceasta,  $v$  este variabilă simplă de tip logic sau componentă de variabilă indexată de tip logic, iar  $e$  este expresie logică.

Executarea ei constă în efectuarea calculelor prevăzute în cadrul expresiei logice după care valoarea logică obținută devine valoare a variabilei logice  $v$ .

c. **Instrucțiunile de atribuire de tip DATA** sînt utilizate pentru inițializări de date, fiind neexecutabile și neetichetabile. Inițializările prevăzute se fac în faza de compilare. Pot figura în program oriunde, cu condiția să nu apară într-o secvență reluabilă de instrucțiuni.

Structura generală este următoarea:

DATA listă de variabile/listă de valori/  
succesiunea

listă de variabile/listă de valori/

putîndu-se repeta, fie punînd virgula după bară înclinată, fie nu.

Într-o listă de valori pot figura variabile simple, componente de variabile indexate, nume de variabilă indexată. În ultimul caz, prezența numelui unei variabile indexate în listă echivalează cu considerarea tuturor componentelor acesteia.



Listele de valori permit, în unele situații scrierea prescurtată. Ca atare, mai multe valori consecutive egale se pot scrie cu factor de multiplicitate, factorul de multiplicitate și valoarea fiind separate prin caracterul stelută (\*). De exemplu, se poate scrie

INTEGER A(5)/5\*2/

în loc de

INTEGER A(5)  
DATA A/5\*2/

#### Observații

1. Valorile din listele de valori asociate lui DATA se pot specifica în unul din modurile acceptate de limbajul FORTRAN, cu respectarea concordanței de tip.
2. Declarațiile pot fi urmate de liste de valori, similar cu DATA, odată cu tipul specificându-se și valorile variabilelor.

#### Exemple

1. Dacă se consideră

INTEGER A,B,C/3\*1/  
LOGICAL D/.TRUE./  
DATA E/'XYZ'/

rezultă că

A=B=C=1

D este variabilă logică cu valoarea adevărată  
B are conținutul XYZb.

2. Fie instrucțiunile:

LOGICAL A,B  
COMPLEX C,D  
DATA A,B,C,D,X/.TRUE.,.FALSE., (1., 1.), (0.,3.), 'T'/

În aceste condiții variabilele din lista lui DATA vor căpăta valorile:  
A=.TRUE.; B=.FALSE.; C este numărul complex 1+i;  
D este numărul complex 3i; X are conținutul Tbbb.

## 4. INSTRUCȚIUNI DE INTRARE/IEȘIRE STANDARD

Sînt instrucțiuni de intrare/ieșire acceptate de orice sistem de calcul, ceea ce conferă generalitate programelor care au ca instrucțiuni de intrare/ieșire astfel de instrucțiuni.

a. **Instrucțiunile FORMAT** se atașează instrucțiunilor de intrare (READ) sau de ieșire (PRINT, PUNCH, WRITE) pentru a asigura conversia informației de pe suport în memorie, respectiv, din memorie pe suport. Forma generală a instrucțiunii este

e FORMAT (d<sub>1</sub>, d<sub>2</sub>, ...)

dacă este vorba de un format atașat unei instrucțiuni de citire, sau este de forma

e FORMAT ('c', d<sub>1</sub>, d<sub>2</sub>, ...)



atunci cînd este atașată unei instrucțiuni de scriere la imprimantă. La ambele forme eticheta  $e$  este obligatorie și este cea specificată în instrucțiunea de citire sau de scriere căreia îi corespunde formatul. După eticheta care se scrie în cîmpul 1—5 al formatului urmează cuvîntul FORMAT scris în cîmpul 7—72 al formularului de programare. În plus, la formatul asociat unei instrucțiuni de scriere la imprimantă apare între ghilimele caracterul de control al scrierii. Tabelul III.1 dă cele patru posibilități de utilizare ale caracterului de control.

TABELUL III.1.

Nr. crt.	Caracterul $c$	Efectul utilizării
1	1	Înainte de scriere se face un salt al hîrtiei imprimantei la o pagină nouă (salt la pagina următoare)
2	0	Înainte de scriere se sare un rînd (scriere la două rînduri)
3	+	Se scrie pe rîndul pe care este poziționat lanțul imprimantei
4	x	Înainte de scriere, se trece la rîndul următor (scrierea la un rînd), $x$ fiind orice caracter afară de cele 3 date (1, 0, +); de obicei se ia blankul pentru scrierea la un rînd

Absența caracterului de control dintr-un FORMAT de scriere nu este semnalată ca eroare. În această situație se ia drept caracter de control primul simbol al primului cîmp descris.

În alcătuirea elementelor  $d_1, d_2, \dots$ , care dau descrierile informației de intrare/ieșire se vor utiliza descriptorii:

I, F, X, A, L, Z

I este destinat datelor de tip întreg, F este destinat datelor de tip real, dublă precizie și complex, A pentru datele alfanumerice, L pentru datele de tip logic, iar X pentru alinieri de date pe suport. Utilizarea celor șase tipuri de descriptori este sintetizată în tabelul III.2.

TABELUL III.2

Nr. crt.	Descriptorul	Cum apare în FORMAT	Efectul
1	I	$nIm$	Describe $n$ cîmpuri de numere întregi, fiecare avînd $m$ poziții (caractere)
2	F	$nFm.p$	Describe $n$ cîmpuri de numere reale, fiecare de cîte $m$ poziții dintre care $p$ sînt după virgulă
3	X	$nX$	Sînt neglijate $n$ poziții. Folosește la alinierea cîmpurilor
4	A	$nAm$	Describe $n$ cîmpuri de tip alfanumeric, fiecare cu $m$ poziții
5	L	$nLm$	Describe $n$ cîmpuri de tip logic, fiecare cu $m$ poziții
6	Z	$nZm$	Describe $n$ cîmpuri de tip număr în baza 16, fiecare de cîte $m$ poziții



Dacă  $n=1$ , atunci descriptorii se scriu respectiv:  $I_m$ ,  $F_m \cdot p$ ,  $1X$ ,  $A_m$ ,  $L_m$ ,  $Z_m$ .

Cînd un grup de descriptori se repetă, se pot grupa între paranteze și în față se scrie factorul de multiplicitate respectiv.

De exemplu, dacă se consideră:

FORMAT (I8, 5X, F7.1, I8, 5X, F7.1, I8, 5X, F7.1)

atunci, condensat, se poate scrie:

FORMAT (3(I8, 5X, F7.1))

Fiecare din descriptori poate fi utilizat fie într-un FORMAT atașat unei instrucțiuni de citire, fie într-unul atașat unei instrucțiuni de scriere. De aceea se vor ilustra cele două ipostaze ale fiecărui descriptor (prin exemple).

#### Observație

În FORMAT se va utiliza și simbolul / care, într-un FORMAT de citire va declanșa trecerea la citirea altei cartele, iar într-un FORMAT de scriere va declanșa trecerea la scrierea altui rînd.

#### Exemple

##### 1. Utilizarea descriptorilor în FORMAT de citire.

Descriptorul	Informația sursă	Ceea ce s-a citit
I3	└12	12
I5	—4321	—4321
F6.2	123456	1234.56
F4.0	└└12	12.
F6.2	└1└23	10.23
A5	ABCDE	BCDE
A3	ABC	ABC└
L1	T	.TRUE.
L2	└F	.FALSE.

##### 2. Utilizarea descriptorilor în FORMAT de scriere.

Descriptorul	Informația sursă	Ceea ce se scrie
I3	12	└12
I5	—4321	—4321
I2	123	**
F6.2	12.	└12.00
F4.0	12.	└12.
F6.3	—123456	*****
A5	ABCD	└ABCD
A3	ABCD	ABC
A4	UWVT	UWVT
L5	.TRUE.	└└└└T
L3	.FALSE.	└└F

#### Observație

Cînd spațiul de scriere nu este suficient pentru a prelua informația sursă, se vor tipări atîtea steluțe cîte s-au prevăzut prin descriptor (vezi exemplele 3 și 6 de mai sus).



b. Instrucțiunile de citire. Un prim tip are forma:

READ ( $p, f$ ) listă

unde:

- READ este cuvântul care arată că este vorba de o instrucțiune de citire (intrare);
- $p$  indică numărul perifericului cu care se face citirea (pentru cititorul de cartele numărul standard este  $p=105$ , la FELIX C—256);
- $f$  este eticheta instrucțiunii FORMAT (cu care se descrie structura listei prevăzută în continuare) atașată instrucțiunii de citire;
- listă este lista variabilelor ale căror valori se precizează.

Mai există și o altă formă a instrucțiunii READ:

READ  $f$ , listă

în care lipsește numărul perifericului ( $p$  de la forma anterioară), înțelegînd prin aceasta că este vorba de cititorul de cartele (aceasta este forma standard).

#### Observație

1. Într-o instrucțiune READ ( $105, f$ ) listă, în loc de etichetă,  $f$  poate fi o zonă de memorie în care, în prealabil s-a introdus formatul de citire, începînd cu paranteza deschisă de după cuvîntul FORMAT și pînă la paranteza finală, inclusiv. Acest lucru oferă un mare avantaj și anume acela că formătul asociat unui READ se poate citi de pe cartele de date, ceea ce conferă programelor o mai mare independență, putîndu-se schimba formatele prin citirea caracterelor de date corespunzătoare.

Dacă de exemplu, se consideră

READ(105,1)A,I,B

1 FORMAT (F5.2, I4,A4)

această secvență se poate înlocui cu următoarea secvență

DIMENSION F(20)

READ 89,F

89 FORMAT(20A4)

READ(105,F)A,I,B

cartela de date asociată primei citiri conținînd

(F5.2,I4,A4)

adică formatul de citire.

2. Poate lipsi lista la o instrucțiune READ

#### Exemple

Dacă READ 1,A,K

1 FORMAT (F4.1,I3)

înseamnă că cele două valori pentru variabilele A și K se vor citi de pe o cartelă din cîmpurile 1—4 și respectiv 5—7 ale acestuia, în caz că nu se utilizează virgula pentru separarea cîmpurilor.

Dacă, de exemplu, se dorește ca:

A = 1.8

K = 4

atunci cartela de date va avea următoarele caractere în primele 7 coloane: —18—4.

Utilizînd virgula ca separator de cîmpuri, cartela va avea în primele 4 coloane următoarele semne: 18,4 de unde rezultă o economie de 3 coloane de cartelă.



Fie instrucțiunea:

```
READ(105,100) (X(K), K = 1,5)
100 FORMAT (5(A4,5X))
```

Cele cinci componente ale variabilei indexate X sînt de tip alfanumeric, fiind citite de pe o cartelă cu 5 cimpuri a căror structură este dată de FORMAT.

c. Instrucțiunile de scriere (WRITE, PRINT, PUNCH). Instrucțiunea WRITE are forma:

$e$  WRITE ( $p, f$ ) listă

unde elementele  $e, p, f$  și listă au aceeași interpretare ca la instrucțiunea READ, numai că  $p=108$  dacă este vorba de scriere la imprimantă sau  $p=106$ , dacă este vorba de perforarea rezultatelor pe cartele (pentru FELIX C-256).

Instrucțiunile PRINT și PUNCH se referă la scrierea la imprimantă și respectiv la perforarea rezultatelor pe cartele. Ele au formele (standard):

PRINT  $f$ , listă  
PUNCH  $f$ , listă

#### Exemple

1. Instrucțiunea:

```
WRITE (108,1) D,A,N
1 FORMAT ('1',5X, I4, 5X, F6.2, 5X, A7)
```

are ca efect scrierea pe primul rînd al unei pagini de imprimantă (efectul caracterului de control 1) a valorilor variabilelor D, A și N. Același lucru se obține scriind:

```
PRINT 1, D, A, N
1 FORMAT ('1', 5X, I4, 5X, F6.2, 5X, A7)
```

2. Dacă vrem ca valorile variabilelor din exemplul anterior să fie perforate fiecare pe cîte o cartelă, atunci vom scrie:

```
WRITE (106,1) D, A, N
1 FORMAT (5X,I4/5X,F6.2/5X,A7)
```

sau

```
PUNCH 1, D, A, N
1 FORMAT (5X,I4/5X,F6.2/5X,A7)
```

În cadrul formatelor se pot utiliza constante alfanumerice. Aceasta, cu scopul creșterii lizibilității.

Dacă, de exemplu, se consideră instrucțiunea:

```
PRINT 4, A, I, V
4 FORMAT ('0', 'A=', F5.1, ' I=', I3, ' V=', A4)
```

atunci se va scrie un rînd la imprimantă (după ce se lasă un rînd liber — efectul lui zero luat drept caracter de control) de forma:

$A=xxx.x \ I=xxx \ V=aaaa$

unde cu  $x$  s-au notat poziții numerice și cu  $a$  poziții alfanumerice.



Dacă se notează cu  $n$  numărul de elemente din lista unei instrucțiuni de scriere și cu  $m$  numărul descriptorilor din formatul asociat (nu se numără descriptorii de tip X și nici constantele alfanumerice) există situațiile:

—  $n < m$  caz în care instrucțiunea se consideră executată la epuizarea listei de variabile (descriptorul  $n + 1$  este neutilizat; să nu se uite că în numărul  $m$  nu se cuprind cei de tip X și constantele alfanumerice);

—  $n = m$  caz în care se epuizează lista odată cu epuizarea formatului;

—  $n > m$  caz în care se reia formatul, de la perechea de paranteze cea mai apropiată de paranteza finală a formatului, pînă cînd se epuizează lista (orice reluare înseamnă o nouă înregistrare) și se ajunge fie la sfîrșitul formatului, fie la  $n + 1$  descriptor.

3. Se dă un text de 160 de caractere scris pe două cartele. Să se scrie un program care să rezolve următoarele probleme:

a) *afișarea textului rezultat* din preluarea caracterului 1, 3, 5, 7, ... 160 ale textului, pe rînduri de cîte 50 de caractere la imprimantă;

b) *scrierea rărită a textului* (între oricare două caractere se inserează un spațiu), rîndurile la imprimantă avînd 40 de caractere;

c) *afișarea inversă a textului*, pe rînduri de 60 de caractere.

*Soluție*

Mai întîi se vor da soluții separate la cele trei probleme și apoi o soluție pentru rezolvarea în ansamblu.

a) Afișarea textului rezultat din prelucrarea caracterelor 1, 3, 5, 7, ... pe rînduri cu cîte 50 de caractere se obține astfel:

```
DIMENSION T(80)
```

```
READ 1, (T(I), I=1,80)
```

```
1 FORMAT (40(A1, 1X)/40(A1, 1X)) → un caracter citit și unul sărit
```

```
PRINT 2, (T(I), I=1,80)
```

```
2 FORMAT (' ', 50 A1)
```

Ca format de citire se putea lua

```
1 FORMAT (40(A1, 1X))
```

b) Scrierea rărită a întregului text este realizată de următoarea secvență de instrucțiuni:

```
DIMENSION T(160)
```

```
READ 1, (T(I), I=1,160) → Citește caracter cu caracter
```

```
1 FORMAT (80A1)
```

```
PRINT 2, (T(I), I=1,160)
```

```
2 FORMAT (' ', 20A2) → Rînduri de 40 caractere (20 de text, 20 spații inserate)
```

c) Afișarea textului de la sfîrșit spre început se realizează cu instrucțiunile:

```
DIMENSION T(160)
```

```
READ 1, T
```

```
1 FORMAT (80A1)
```

```
PRINT 2, (T(160-I+1), I=1,160)
```

```
2 FORMAT (' ', 60A1)
```

O variantă a unui program care rezolvă toate cele trei probleme este dată în figura III.1 unde s-au completat și cartelele de date (două cartele) și cartelele de comandă așa încît programul să poată fi rulat pe un calculator FELIX C-256 (cartelele de comandă au punct în coloana 1 și se scriu începînd din coloana 11).







4. Să se scrie un program pentru a calcula produsul a patru numere întregi ale căror valori se citesc de pe câte o cartelă din câmpul 7—12. Se cere să se afișeze rezultatul pe primul rând al unei pagini noi și apoi, după un rând liber, să se afișeze pe patru rânduri fiecare din cele patru valori, în ordinea inversă citirii, sub forma

nume = valoare

începînd din coloana 10 a rîndului respectiv.

Primul rînd se cere să aibă forma

PRODUSUL CELOR PATRU NUMERE ESTE valoarea, și să fie scris începînd din coloana 1 a rîndului.

#### *Soluție*

Fie A, B, C, D numele celor patru variabile necesare și R variabila pentru rezultat.

Ținînd cont de cerințele problemei și de notațiile alese, rezultă programul:

```

INTEGER A, B, C, D, R
READ 1, A, B, C, D
1 FORMAT (6X, I6)
  PRINT 2, R, D, C, B, A
2 FORMAT ('1', 'PRODUSUL CELOR PATRU NUMERE ESTE',
  * I8/'0', 9X, 'D=', I6/' ', 9X, 'C=', I6/' ', 9X, 'B=', I6/' ', 9X,
  * 'A=', I6)
  STOP

```

Se observă că rezultatul s-a scris cu descriptorul I8, pe pagină nouă, iar D, C, B, A s-au scris cu descriptorul care cu s-au citit (I6), fiecare pe câte un rînd.

De asemenea, se remarcă faptul că instrucțiunea FORMAT de etichetă 2 s-a continuat pe alt rînd marcat cu \* în coloana 6 (se putea utiliza orice alt caracter afară de zero sau blank); faptul că nu s-a completat rîndul de început al formatului pînă în coloana 72 nu constituie o eroare deoarece (reamintim) spațiul nu este semnificativ pentru compilatorul FORTRAN.

5. Se dau variabilele logice A, B, C, D, variabilele E, F, G de tip dublă precizie și variabilele H, I, J de tip complex. Să se determine:

$$\begin{aligned}
 A &= B \wedge \bar{C} \vee \bar{B} \wedge \bar{D} \\
 E &= F + G \\
 H &= I \cdot J
 \end{aligned}$$

Valorile pentru B, C, D, F, G, I, J se vor preciza printr-o instrucțiune de tip DATA, iar rezultatele se vor scrie pe un același rînd la imprimantă, separate de câte două spații.

#### *Soluție*

În condițiile de mai sus, programul FORTRAN de soluționare a problemei este următorul:

```

LOGICAL A, B, C, D
DOUBLE PRECISION E, F, G
COMPLEX H, I, J

```



```

DATA B, C, D, F, G, I, J/'T', 'F', .FALSE., 0.5D2,5.,
*(10.54,19.),(.0,75.)/
A=B.AND..NOT.C.OR..NOT.B.AND..NOT.D
E=F+G
H=I*J
PRINT 1, A, E, H
1 FORMAT (' ', A = ', L1, 2X; 'E=', F7.2, 2X, 'H=', 2F6.1)
STOP

```

### INTREBĂRI ȘI PROBLEME

1. Care sînt tipurile de instrucțiuni de atribuire?
2. Dacă  $I = 3$ ,  $J = 2$ ,  $A = 1$ ,  $B = 3.6$ ,  $X = (1, 2.5)$ ,  $Y = (0., 5.)$ , să se spună ce efect au instrucțiunile:

```

L = I ** J ** I ** 2/I/J*2
C = 2. * I/J * 3 - A
X = X/Y * X
V(1.5 * J - 2) = J + I/2 + I/2

```

știind că L, C, V sînt variabile de tip întreg, real, real.

3. Fie  $I = 2$ ,  $K = 4$ ,  $L = -1$ ,  $V = -1.$ ,  $B = .TRUE.$ ,  $C = .FALSE.$ ,  $D = .FALSE.$  și X, Y, Z variabile de tip logic. Să se spună ce efect au instrucțiunile

```

X = .NOT. B .OR. C .AND. D .AND. .NOT. B
Y = K * I .LE. V * B .AND..NOT.C.OR.I**K.NE.1
Z = .NOT..NOT.B.OR..NOT.C.AND..NOT.D

```

4. Fie N variabilă de tip alfanumeric cu conținutul ABCD. Ce conținut (alfanumeric) au A și I dacă se efectuează instrucțiunile

A=N și I=N

5. Să se indice valorile citite în cazurile ce urmează.

Descriptor	Informația sursă	Descriptor	Informația sursă
I4	b4bb	F5.2	b1b.b
F3.0	-1.	F4.2	1.5
A5	12345	A3	1AB,
L6	bbbbTT	L3	ABC

6. Să se indice valorile scrise în cazurile de mai jos.

Descriptor	Informația sursă	Descriptor	Informația sursă
I4	4000	I5	123456
F5.2	100.	F4.0	-123.5
A3	XYZT	A5	ABCD
L3	.TRUE.	L2	.FALSE.

7. Să se spună câte cartele de date sînt necesare pentru fiecare dintre citirile care urmează:

```

READ 1000, A, I, B, J, K
1000 FORMAT (F4.0, I4)
READ 1, AN, LU, ZI, BA, CA, DA, EA
1 FORMAT (5X, A3/I4, 2(1X, F5.0), F3.1)
READ 3, (X(I), I = 1, 10, 3)

```



```

3  FORMAT (F1.0/2(F3.0, 1X, F6.2)/F3.2/)
   READ (105,1)(A(I+2), I=1,7,2)
1  FORMAT (/A1/A2)

```

8. Să se indice câte rînduri la imprimantă, respectiv, câte cartele, necesită rezultatele în cazul instrucţiunilor care se dau în continuare.

```

      PRINT 1,A,I,J,K
1  FORMAT ('O', A3, I4/(' ', 15))
      WRITE (108,2) (I(L), L=2,7,4)
2  FORMAT ('1', I3,(I1), I4/)
      PUNCH 3,P,A,R,D,O,N
3  FORMAT (' ', 'VALORI: '/(' ',2(F3.0,A3)))

```

9. Fie hABCDEFCHITF 12345678 conţinutul primelor 20 de coloane ale unei cartele. Ce se va scrie la imprimantă şi sub ce formă dacă se execută secvenţa de program:

```

LOGICAL A,B
PRINT 1
FORMAT ('1', 'PROGRAMARE', 2I1,2I4, 'STOP')
READ 1,A,B,I,K
PRINT 1,A,B,I,K

```



## CAPITOLUL IV

### INSTRUCȚIUNI PENTRU SCRIEREA PROGRAMELOR CU STRUCTURĂ ALTERNATIVĂ

#### 1. INSTRUCȚIUNI DE TIP GO TO

Limbajul FORTRAN acceptă trei categorii de *instrucțiuni* de tip **GO TO** și anume:

GO TO  $e$   
GO TO  $v, (i_1, i_2, \dots, i_m)$   
GO TO  $(i_1, i_2, \dots, i_m), v$

numite, respectiv, transfer necondiționat, transfer impus cu un **ASSIGN** și transfer calculat. Prin  $e$  s-a notat o etichetă FORTRAN, cu  $v$  o variabilă simplă de tip întreg, iar  $i_1, i_2, \dots, i_m$  sînt etichete FORTRAN.

O instrucțiune de forma

GO TO  $e$

are ca efect trecerea (necondiționată) la efectuarea instrucțiunii de etichetă  $e$ .

Orice instrucțiune de-al doilea tip

GO TO  $v, (i_1, i_2, \dots, i_m)$

este precedată (logic) de o instrucțiune

ASSIGN  $i$  TO  $v$

unde  $i$  este una din valorile  $i_1, i_2, \dots, i_m$ . În acest fel, instrucțiunea **GO TO** devine echivalentă cu

GO TO  $i$

Al treilea tip, **GO TO** calculat, a rezultat ca o perfecționare adusă lui **GO TO** impus cu **ASSIGN** și se execută astfel: după ce se atribuie lui  $v$  o valoare  $r$  (una din valorile  $1, 2, \dots, m$ ), efectul este **GO TO**  $r$  (se trece la executarea instrucțiunii de etichetă  $r$ )

#### Observație

1. La unele compilatoare virgula ce separă variabila  $v$  de lista de etichete este opțională.
2. În cele ce urmează se va utiliza mai mult **GO TO** necondiționat.
3. La **GO TO** calculat pot exista etichete egale în lista de etichete.
4. În cazul că lipsește lista la **GO TO** cu **ASSIGN** instrucțiunea devine: **GO TO**  $r$

#### Exemple

```
GO TO 1937
ASSIGN 3 TO K
GO TO K, (4, 3, 7, 1)
      ↑
      K = 3

I = 2
L = 4 * I ** 2 - 3
```



GO TO (4, 1, 90, 51, 6, 7, 3, 8, 8, 8, 8, 8, 8), L  
 $\uparrow$   
 L=13

ASSIGN 13 TO M  
 GO TO M

## 2. INSTRUCȚIUNI DE TIP IF

Există două feluri de instrucțiuni de acest tip.

a. Instrucțiunea IF aritmetic are următoarea formă generală:

IF (*ea*) *k1*, *k2*, *k3*

în care:

- *ea* este o expresie aritmetică fără operanzi de tip complex;
- *k1*, *k2*, *k3*; sînt etichete.

O instrucțiune IF aritmetic se execută astfel:

- se evaluează expresia *ea*;
- dacă valoarea rezultată este strict negativă, se va trece la executarea instrucțiunii cu etichete *k1* (efectul: GO TO *k1*);
- dacă valoarea expresiei este egală cu zero se va trece la executarea instrucțiunii cu eticheta *k2* (efectul: GO TO *k2*);
- dacă expresia are o valoare strict pozitivă se va executa instrucțiunea cu eticheta *k3* (efectul: GO TO *k3*).

Dacă se iau  $k1=k2$  înseamnă că atunci cînd valoarea expresiei este mai mică sau egală cu zero se va trece la același punct din program, iar în caz contrar, se execută instrucțiunea cu eticheta *k3*. Așadar, există posibilitatea ca două din cele trei etichete scrise după IF să fie egale, ceea ce permite realizarea diverselor condiții cerute în rezolvări de probleme ( $\geq$ ,  $\leq$ ,  $\neq$ ).

### Observație

Orice instrucțiune care urmează unui IF aritmetic sau unui GO TO trebuie să fie etichetată pentru că altfel nu ar fi accesibilă niciodată: nu este greșit dacă se iau  $k1=k2=k3$  dar, în acest caz, IF este inefficient, avînd rol de GO TO necondiționat.

b. Instrucțiunea IF logic are următoarea formă generală:

IF (*el*)*i*

unde:

- *el* este o expresie logică;
- *i* este o instrucțiune, alta decît un IF logic sau o instrucțiune DO (instrucțiunea DO se va prezenta în capitolul următor).

O astfel de instrucțiune se execută după cum urmează:

- se evaluează expresia logică *el*;
- dacă valoarea logică rezultată este .TRUE., atunci se execută instrucțiunea *i*, iar dacă valoarea logică este .FALSE. se execută instrucțiunea imediat următoare lui IF logic.

### Exemple

1. IF (A)110, 20, 300  
 IF (A\*X-C)1, 1, 22



```

13 IF ((T-A)**2-(C+D)**2)40, 40, 50
   IF (T.GE.0)GO TO 7
   IF ((A+B+C).GT.100) PRINT 1, B, C
1  FORMAT ('0', 3(F6.2, 3X))

```

2. Să se scrie programul pentru a determina pe MAX astfel:

MAX=max {A, B, C}

MAX, A, B, C, fiind numere întregi cu câte cel mult cinci cifre fiecare.

*Soluție*

Presupunem că A se citește de pe o cartelă, din câmpul 1—5, iar B și C se citesc de pe o a doua cartelă din câmpurile 2—6 și respectiv 7—11 iar rezultatul trebuie să se scrie pe două rînduri la imprimantă sub forma:

```

      CEL MAI MARE DINTRE A, B, C și apoi valoarea pe alt rînd.
      INTEGER A, B, C
      READ 1, A, B, C
1  FORMAT (I5/1X,I5,1X,I5)
      MAX=A
      IF(MAX.LT.B)MAX=B
      IF (MAX.LT.C)MAX=C
      PRINT 2, MAX
2  FORMAT (' ', 'CEL MAI MARE DINTRE A, B, C: '/7X, I5)
      STOP

```

### 3. SIMULAREA STRUCTURILOR IF-THEN, IF-THEN-ELSE, CASE-OF

Structurile de tip **IF-THEN** se pot genera în două moduri, cu sau fără inserarea unui comentariu, astfel:

<pre> IF(.NOT.c) GO TO 11111 C      THEN </pre>	<pre> IF(.NOT.c) GO TO 22222 </pre>
---	-------------------------------------

instrucțiuni

instrucțiuni

11111 CONTINUE

22222 CONTINUE

c fiind condiția care determină structura (etichetele pot fi altele).

*Exemple*

<pre> IF(.NOT.A.LE.0) GO TO 12345 C      THEN       PRINT 1,A 1  FORMAT('0',A15) 12345 CONTINUE </pre>	<pre> IF(.NOT.X.GT.Y) GO TO 37       S=A+B/2       T=B+A/2       W=(A+B)/2 37 CONTINUE </pre>
--	---

Structurile de tip **IF-THEN-ELSE** se vor simula în felul următor:

```

IF(.NOT. c) GO TO 15378
C      THEN

```

instrucțiuni

```

GO TO 98765
C      ELSE

```



## 15378 CONTINUE

instrucțiuni

## 98765 CONTINUE

Exemplu

```

DATA A, B, C/3,0.,2./
X=A+B+C
IF(.NOT.B.C.NE.0) GO TO 11111
C      THEN
      Y=A/B+1C
      PRINT 22222, X, Y
22222  FORMAT('1', 'X=', F5.1, 'Y=', F5.1)
      GO TO 33333
C      ELSE
11111  CONTINUE
      X=A*A*A
      PRINT 44444, X
44444  FORMAT('0', 'X=', F5.1, 'Y NU ARE SENS')
33333  CONTINUE

```

Structurile de tip CASE-OF se simulează astfel:  
 GO TO ( $i_1, i_2, \dots, i_n$ ),  $e$

$i_1$  instrucțiuni

GO TO  $e$

$i_2$  instrucțiuni

GO TO  $e$

$i_n$  instrucțiuni

$e$  CONTINUE

3. Se dau A, B, C, D numere reale. Să se determine valoarea R astfel:

$$R = \begin{cases} A & \text{dacă } N=1 \\ A+B & \text{dacă } N=2 \\ A+B+C & \text{dacă } N=3 \\ A+B+C+D & \text{dacă } N=4 \end{cases}$$

Soluție

Se va utiliza un GO TO calculat care va avea ca variabilă de control chiar pe N. Valoarea lui N se va citi odată cu cele ale lui A, B, C, D. Pentru siguranța că se citește o valoare corectă pentru N se va face verificarea pentru a ști dacă N ia una din cele 4 valori (1, 2, 3, 4).



```

          READ 98765, A, B, C, D, N
98765     FORMAT(4F5.2, I5)
          IF(.NOT.((N.GE.1).AND.(N.LE.4))) GO TO 11111
C         THEN
          GO TO (10, 11, 12, 13), N
          10      R=A
              TO TO 22222
          11      R=A+B
              GO TO 22222
          12      R=A+B+C
              GO TO 22222
          13      R=A+B+C+D
22222     CONTINUE
          PRINT 66666, R
66666     FORMAT('1', F7.2)
          GO TO 12345
C         ELSE
11111     CONTINUE
          PRINT 88888
88888     FORMAT('O', 'N NU IA UNA DIN CELE PATRU VALORI')
12345     CONTINUE
          STOP

```

### PROBLEME

1. Se dă un text scris pe 10 cartele. Să se afișeze ce conține a 10-a cartelă dacă primul caracter al textului este litera S și să se afișeze conținutul primei cartele în caz contrar.
2. Dându-se textul menționat la problema anterioară, să se numere de câte ori apare litera J în întregul text.
3. Se dau trei cartele pe care sînt scrise cîte 20 de valori de tip întreg, în cîmpuri de cîte patru coloane. În fața acestor cartele este o altă cartelă pe care, în coloanele 75–76, este dat un număr natural  $N(N \leq 60)$ . Să se calculeze suma numerelor strict pozitive dintre primele  $N$  numere date.
4. Pe o cartelă, în cîmpuri de cîte două coloane, se dau 40 de valori de tip logic. O următoare cartelă conține două valori reale  $A, B$ , respectiv în cîmpurile 5–10, 20–25, cu două cifre în partea fracționară. Să se calculeze:

$$S = \begin{cases} A + B & \text{dacă vectorul logic are cel puțin 30 de componente cu valoarea TRUE.} \\ A \cdot B & \text{în caz contrar} \end{cases}$$

Se vor afișa atît componentele vectorului logic cît și valoarea  $S$ , toate pe un singur rînd la imprimantă.

5. Fiind date  $A, B, C, D$  variabile reale, să se permute valorile lor astfel încît:

$$A \leq B \leq C \leq D$$

6. Dacă  $A, B, C, D, E$  sînt variabile booleene, să se arate dacă au valorile alternative (0, 1, 0, 1, 0 sau 1, 0, 1, 0, 1).
7. Fiind date  $A, B, C, D$  numere întregi diferite de zero, cu cel mult cinci cifre, să se spună dacă unul singur este negativ și celelalte trei sînt pozitive, fără a aplica un procedeu de numărare.



8. Fiind dat vectorul:

$$V(1), V(2), \dots, V(N)$$

de tip întreg, cu  $N \leq 100$ , să se mute eventualele elemente nule la sfârșit, fără a utiliza un al doilea vector. Dacă, de exemplu,  $V$  ar fi  $(7, 0, -2, 4, 3, 0, 0, 1)$  atunci el trebuie să devină  $(x, x, x, x, x, 0, 0, 0)$  cu  $x \neq 0$ .

9. Fie  $A(N, N)$  o matrice cu elemente numere naturale. Presupunind că  $N \leq 15$ , să se afle suma elementelor de sub diagonala principală.
10. Tot pentru matricea considerată la exercițiul anterior, să se calculeze suma elementelor marginale (determinate de linia 1, linia  $N$ , coloana 1, coloana  $N$ ).
11. Se dă un text scris pe 80 de cartele. Să se afișeze la imprimantă, pe un rând, ceea ce rezultă prin preluarea caracterelor 1 din cartela 1, 2 din cartela 2, 3 din cartela 3, ..., 80 din cartela 80.
12. Se dă un șir de 80 de cifre scrise pe o cartelă. Să se spună de câte ori se întâlnește numărul 1978.
13. Dându-se un șir de  $80 \times 80$  cifre scris pe 80 de cartele, să se afișeze numărul rezultat prin preluarea cifrelor 1 din cartela 1, 80 din cartela 2, 1 din cartela 3, 80 din cartela 4, ...
14. Se dau opt cifre ale sistemului zecimal scrise fiecare pe câte o cartelă, în coloana 80. Pentru fiecare cifră să se afișeze cuvântul ce o desemnează în limba română (respectiv, UNU, DOI, ...).
15. Se dă un număr natural de 10 cifre, pe o cartelă, în coloanele 11–20. Să se spună dacă numărul este mai mare sau egal cu 100 000.



## CAPITOLUL V

### INSTRUCȚIUNI PENTRU DESCRIEREA STRUCTURILOR REPETITIVE

În funcție de tip, *structurile repetitive* pot fi descrise în limbajul FORTRAN fie prin unele din instrucțiunile studiate, fie printr-o instrucțiune specifică.

#### 1. SIMULAREA STRUCTURII REPETITIVE DE TIP DO-WHILE

Forma generală a unei *structuri* repetitive de tip **DO-WHILE** este dată în figura V.1.

Structura prezentată se descrie în limbajul FORTRAN astfel:

```

e1  -----
    CONTINUE
    IF( $\bar{c}$ ) GO TO e2
                                secvența s
                                GO TO e1
e2  -----
    CONTINUE
    
```

de unde rezultă că simularea unei structuri repetitive de tip DO-WHILE în FORTRAN, se realizează prin instrucțiunile: IF *logic*, GO TO, CONTINUE.

În structura prezentată în figura V.1 se observă că execuția secvenței de instrucțiuni *s* are loc atunci când propoziția logică *c* este adevărată. Din acest motiv în instrucțiunea IF propoziția logică este negată ( $\bar{c}$ ), iar când aceasta este adevărată (deci propoziția logică *c* este falsă) se va executa instrucțiunea CONTINUE de etichetă *e2* al cărei efect este trecerea la instrucțiunea imediat următoare, secvența *s* nemaie executându-se.

Ori de câte ori propoziția logică  $\bar{c}$  este falsă (deci propoziția logică *c* este adevărată) se va executa secvența de instrucțiuni *s*, datorită instrucțiunii CONTINUE de etichetă *e1*, la care are loc trimiterea prin instrucțiunea GO TO plasată imediat după secvența *s*.

Se remarcă rolul instrucțiunii CONTINUE în evidențierea procesului de execuție repetată a secvenței de instrucțiuni *s*, fiind singura instrucțiune care este etichetată.

*Programul V.1.* Să se efectueze împărțirea a două numere naturale prin scăderi succesive.

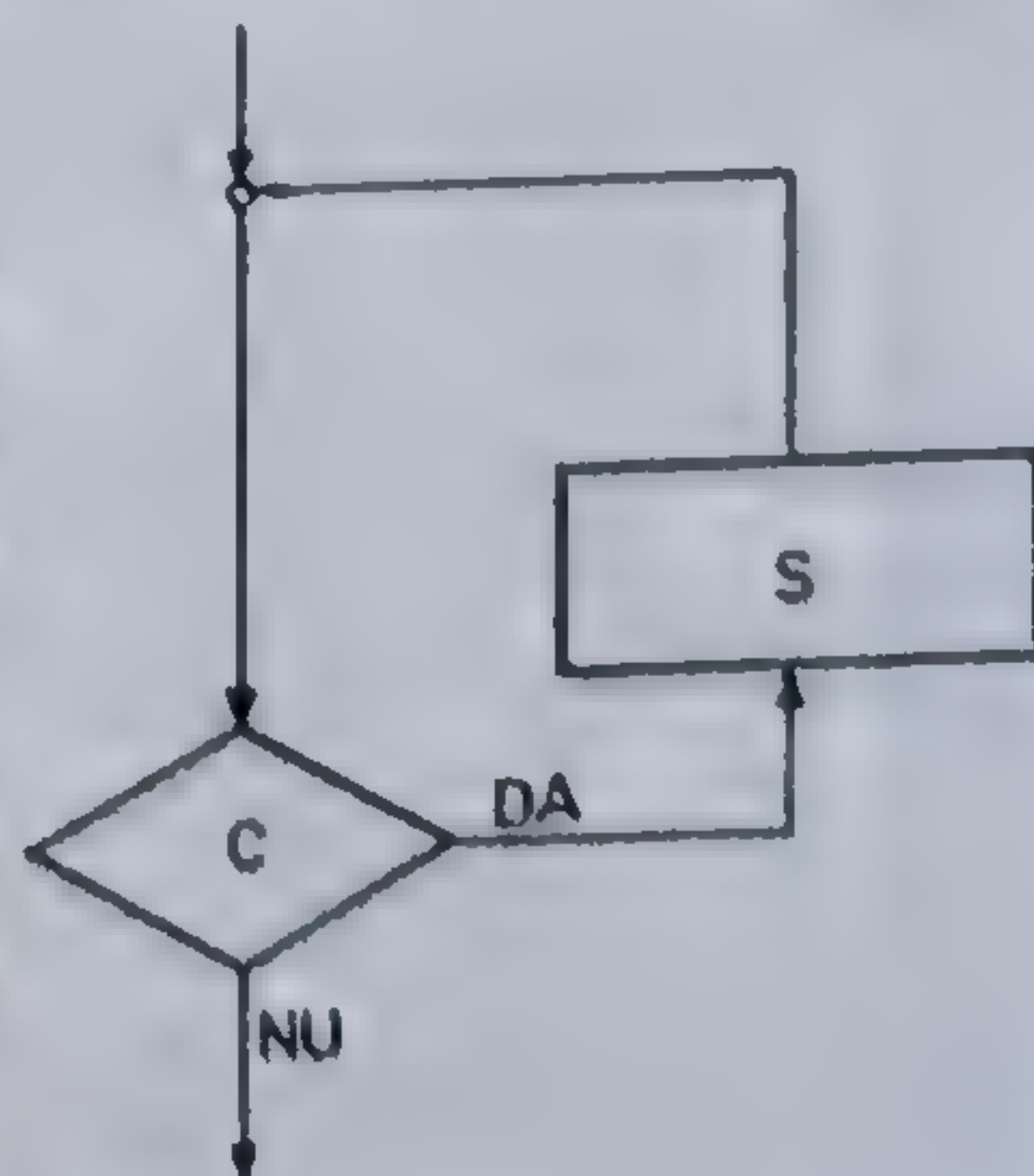


Fig. V.1.



```

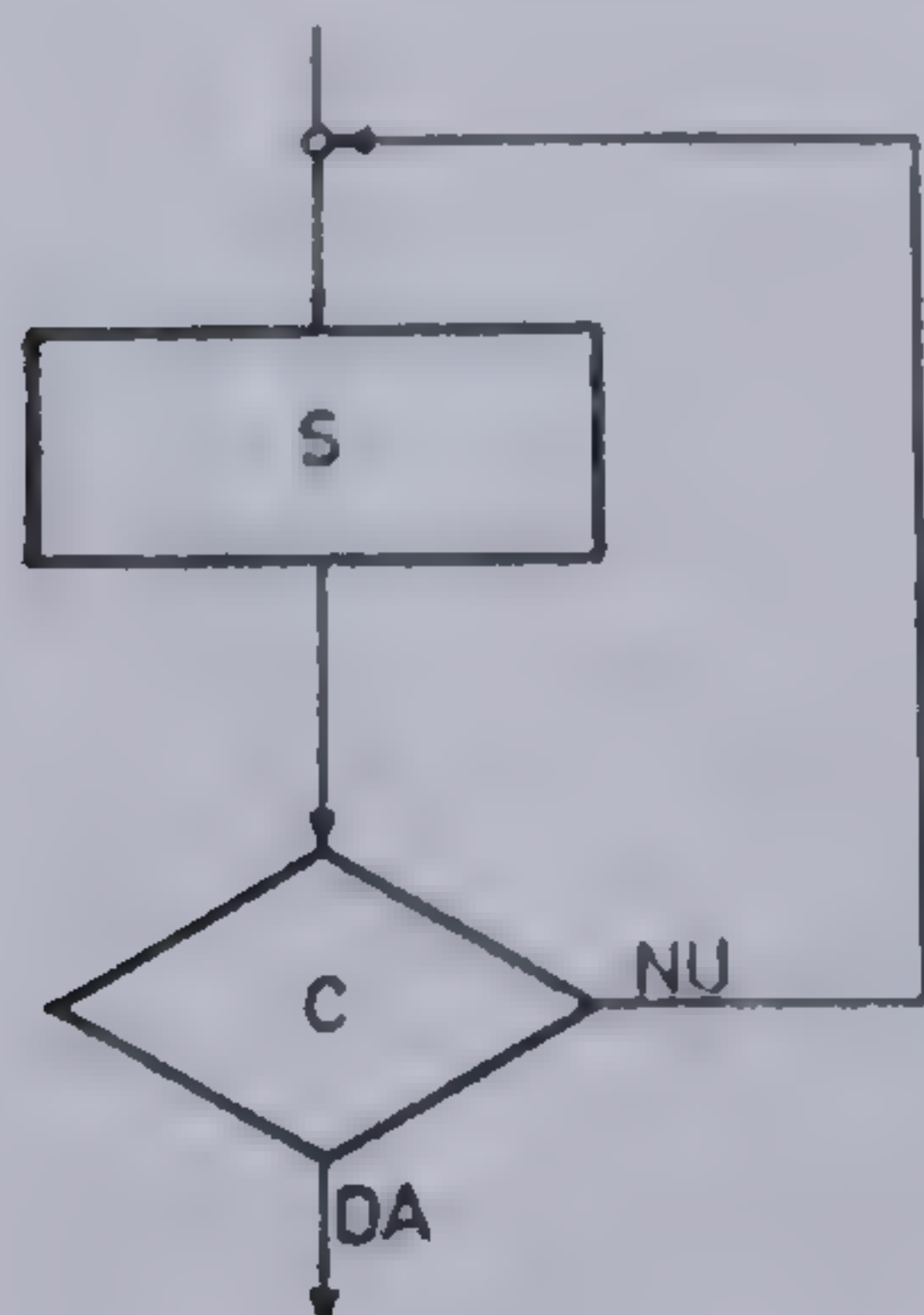
•      JOH PR51,AN:1234,PN:MIHAI
•      COMPILE FORTRAN
C
C      *****
C      * SIMULAREA UNEI STRUCTURI REPETITIVE *
C      *   DE TIP DO-WHILE                     *
C      *****
C
      INTEGER Q,D
      READ(105,1) M,N
1  FORMAT(2I3)
      Q=0
C      INCEPUT SIMULARE STRUCTURA DO-WHILE
2  CONTINUE
      IF(M.LT.N) GO TO 3
          D=M-N
          Q=Q+1
          M=D
          GO TO 2
3  CONTINUE
C      SFIRSIT SIMULARE STRUCTURA DO-WHILE
      WRITE (108,4) Q,M
4  FORMAT('  ',Q=' ',I3,'M=' ',I2)
      STOP
      END.
•      LINK
•      RUN
742 48
      EOJ  PROGRAMUL V.1

```

## 2. SIMULAREA STRUCTURII REPETITIVE DE TIP DO-UNTIL

Forma generală a unei *structuri* repetitive de tip **DO-UNTIL** este dată în figura V.2.

Structura prezentată se descrie în limbajul FORTRAN astfel:



```

.....
e1  CONTINUE
                                [ secvența s ]
.....
e2  IF(c) GO TO e1
    CONTINUE
.....

```

Se observă că simularea unei structuri repetitive de tip DO-UNTIL se realizează, ca și cea anterioară, prin instrucțiunile: IF *logic*, CONTINUE, GO TO.

Fig. V.2.



Deoarece în structura prezentată în figura V.2 execuția secvenței de instrucțiuni s are loc atunci când propoziția logică  $c$  este falsă, în instrucțiunea IF propoziția logică este negată ( $\bar{c}$ ), astfel încât atunci când aceasta este adevărată (deci propoziția logică  $c$  este falsă) să se execute secvența de instrucțiuni s, iar atunci când propoziția logică  $\bar{c}$  este falsă (deci  $c$  este adevărată) să se execute instrucțiunea CONTINUE de etichetă  $e_2$ , ceea ce conduce la instrucțiunea imediat următoare.

Programul V.2. Să se ordoneze crescător trei numere naturale.

```

•      JOB PR52,AN:1234,PN:MIHAI
•      COMPILE FORTRAN
C
C      -*****
C      *   SIMULAREA UNEI STRUCTURI REPETITI- *
C      *   VE DE TIP DO-UNTIL                  *
C      *****
C
      INTEGER A,B,C,AUX
      READ(105,1) A,B,C
1  FORMAT(3I3)
C      INCEPUT SIMULARE STRUCTURA DO-UNTIL
2  CONTINUE
      K=0
      IF(A.LE.B) GO TO 3
      K=1
      AUX=A
      A=B
      B=AUX
3      CONTINUE
      IF(B.LE.C) GO TO 4
      K=1
      AUX=B
      B=C
      C=AUX
4      CONTINUE
      IF(K.NE.0) GO TO 2
5  CONTINUE
C      SFIRSIT SIMULARE STRUCTURA DO-UNTIL
      WRITE (108,6) 'A,B,C'
6  FORMAT(' ', 'A=',I3, 'B=',I3, 'C=',I3)
      STOP
      END
      LINK
      RUN
•      89 72 54
•      EOJ

```

PROGRAMUL V.2



### 3. SIMULAREA STRUCTURII REPETITIVE CONDIȚIONATĂ INTERN

Forma generală a unei structuri repetitive condiționată intern **LOOP-EXITIF-ENDLOOP** este dată în figura V.3.

Structura prezentată se descrie în limbajul FORTRAN astfel:

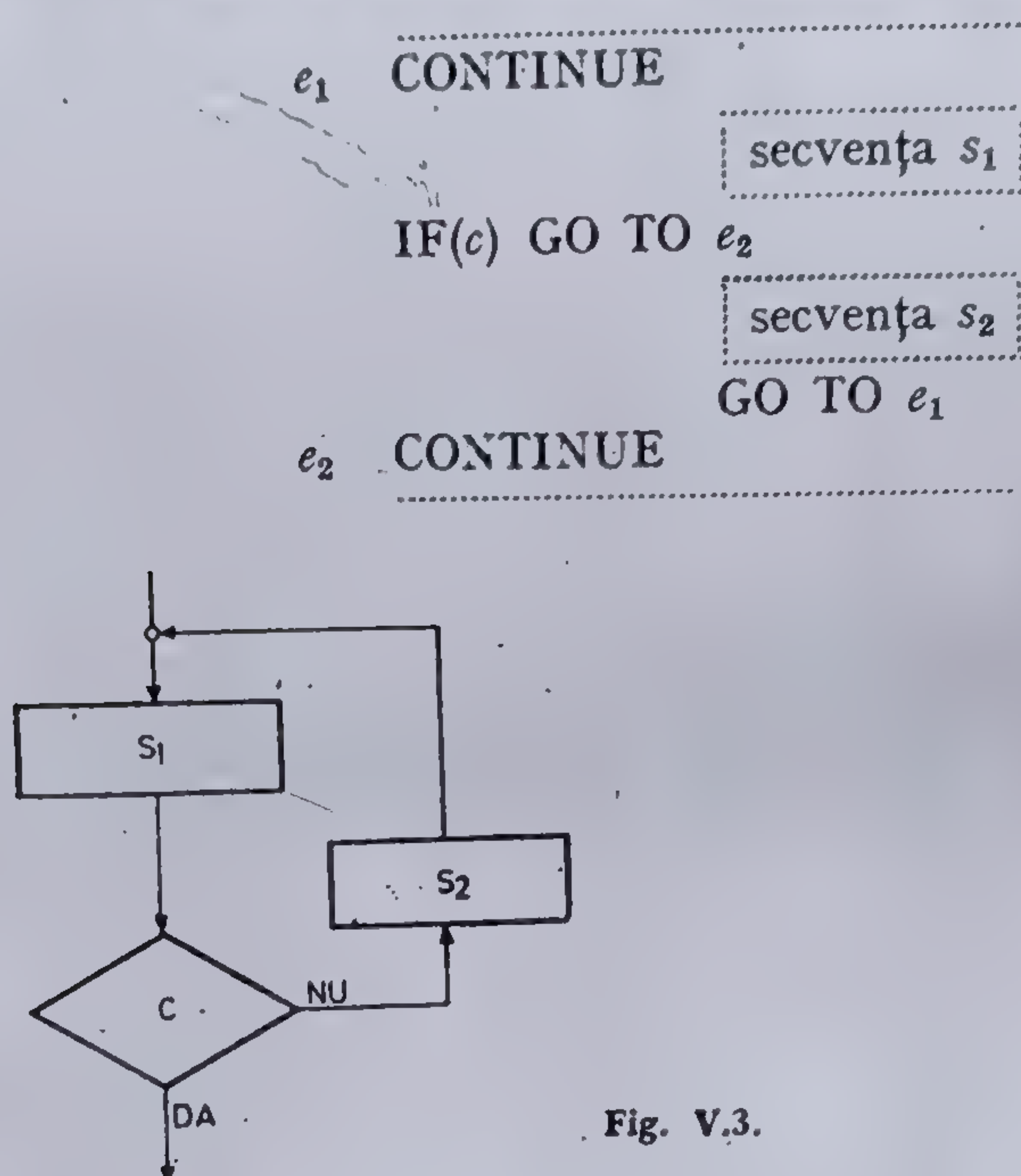


Fig. V.3.

Se observă că simularea unei structuri repetitive condiționată intern se realizează prin instrucțiunile: *IF logic*, *CONTINUE*, *GO TO*.

Condiția de terminare a execuției structurii repetitive se află în interiorul secvenței de instrucțiuni care se repetă, astfel încât rezultă o subsecvență  $s_1$  condiționată posterior (structură DO-UNTIL) și alta  $s_2$  condiționată anterior (structură DO-WHILE).

*Programul V.3.* Să se afle cel mai mare divizor comun a două numere naturale  $M$  și  $N$  prin algoritmul lui Euclid.

#### Observație

O structură repetitivă mai poate fi denumită *structură ciclică* sau pe scurt *ciclu*.

Se acceptă, deci, să se înțeleagă prin *ciclu*, un lanț complet de operații care se execută într-o anumită secvență, a algoritmului și care prin repetare epuizează evoluția unui anumit proces de calcul.

Secvența de instrucțiuni care se repetă în execuție se numește *corpul ciclului*, iar instrucțiunea *IF logic* plasată înaintea, după sau în interiorul secvenței, permite ieșirea din ciclu.

În toate cele trei tipuri de structuri repetitive studiate, secvența de instrucțiuni (*corpul ciclului*) se repetă de un număr necunoscut de ori (determinat de condiția  $c$ ).



```

      JOB PR53,AN:1234,PN:MIHAI
      COMPILE FORTRAN
      *****
      *   SIMULAREA UNEI STRUCTURI REPETITIVE   *
      *   DE TIP LOOP-EXITIF-ENDLOOP           *
      *****

      INTEGER Q,R,AUX,CMMDC
      READ(105,1) M,N
1  FORMAT(2I3)
      IF(M.GE.N) GO TO 2
          AUX=M
          M=N
          N=AUX
2  CONTINUE
      INCEPUT SIMULARE STRUCTURA
      LOOP-EXITIF-ENDLOOP
          Q=M/N
          R=M-N*Q

          IF(R.EQ.0) GO TO 3
              M=N
              N=R
              GO TO 2
3  CONTINUE
      SFIRSIT SIMULARE STRUCTURA
      LOOP-EXITIF-ENDLOOP
      CMMDC=N
      WRITE(108,4) CMMDC
4  FORMAT(' ',CMMDC=' ',I3)
      STOP
      END

      LINK
      RUN
385 80
      EOJ

```

PROGRAMUL V.3

#### 4. DESCRIEREA UNEI STRUCTURI REPETITIVE DE TIP DO-FOR

Forma generală a unei *structuri* repetitive de tip DO-FOR este dată în figura V.4.

Se observă că grupul de instrucțiuni ce constituie secvența s (corpul ciclului) se execută de un număr dat de ori.



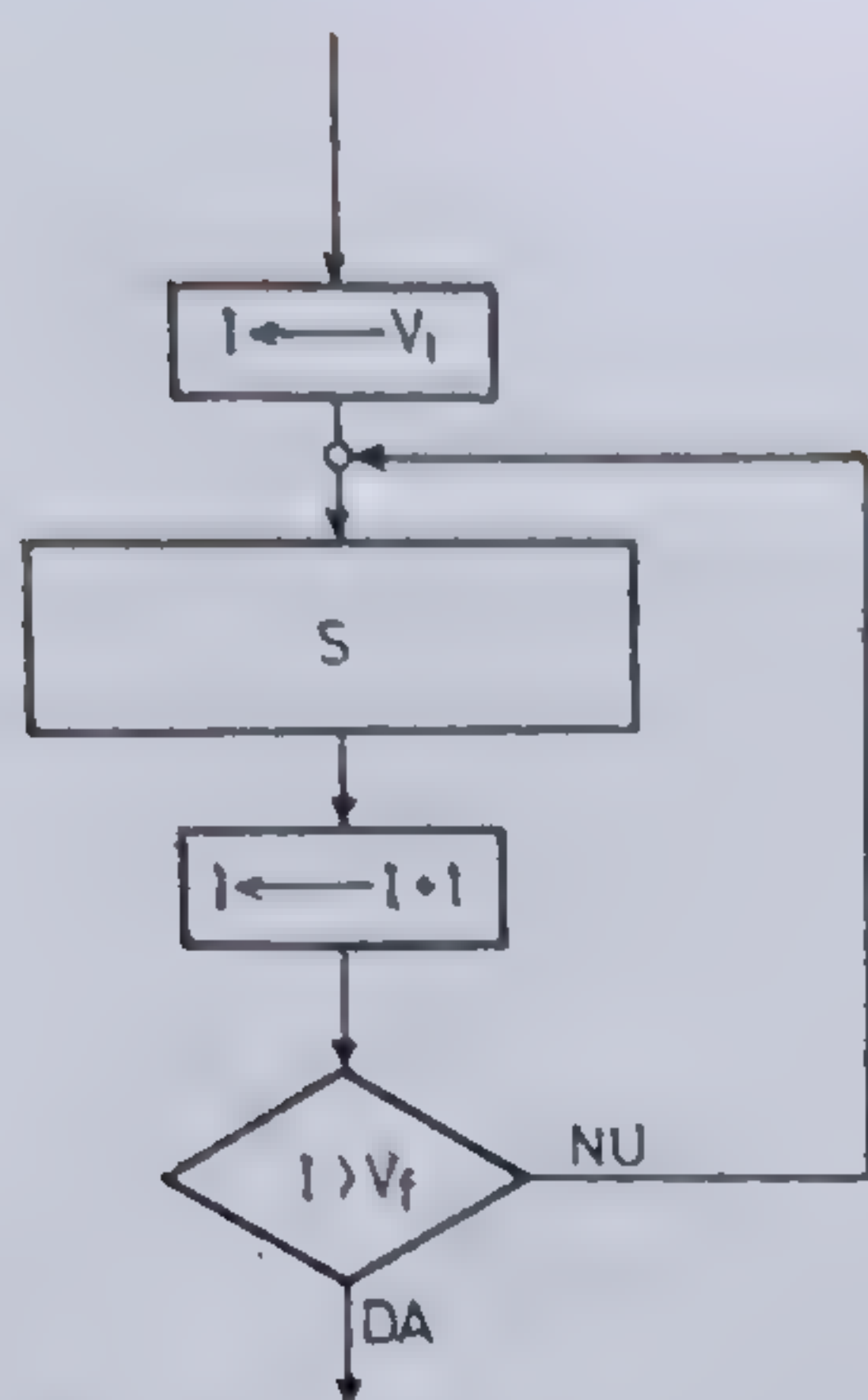


Fig. V.4. Structura repetitivă DO-FOR

Instrucțiunea FORTRAN, care permite executarea operațiilor din corpul unui ciclu de un număr cunoscut de ori, este *instrucțiunea DO*.

Format general

$[e]DO\ k\ I = M1, M2, M3$

unde:

— eticheta  $e$  se utilizează atunci când există o referire la această instrucțiune;

—  $k$  este eticheta unei instrucțiuni executabile, ultima din grupul de instrucțiuni care se repetă, numită instrucțiune terminală; instrucțiunea terminală nu poate fi una din instrucțiunile următoare: GO TO, IF aritmetic, RETURN, STOP, IF logic conținând una din instrucțiunile precedente și DO (de regulă, drept instrucțiune terminală, se va utiliza instrucțiunea CONTINUE);

—  $I$  este numele unei variabile întregi, numită variabilă de control a ciclului;

—  $M1, M2, M3$  sînt, fiecare, fie o constantă întreagă, pozitivă fie un nume de variabilă întreagă, reprezentînd respectiv:

$M1$  valoarea inițială,

$M2$  valoarea finală,

$M3$  valoarea de avans (rația).

Dacă  $M3$  nu apare în formatul instrucțiunii DO înseamnă că  $M3=1$ .

*Secvența unei instrucțiuni DO.* Fiecărei instrucțiuni DO îi este asociată o secvență definită de ansamblul tuturor instrucțiunilor executabile corespunzătoare operațiilor din corpul ciclului, cuprinse între prima instrucțiune executabilă din corpul instrucțiunii DO și instrucțiunea terminală. Atît prima instrucțiune, cît și instrucțiunea terminală, fac parte din secvența instrucțiunii DO.

În executarea unei instrucțiuni DO sînt parcurse următoarele faze:

— variabilei de control  $I$  îi este atribuită valoarea inițială  $M1$ ;

— se execută secvența asociată instrucțiunii DO;

— valoarea variabilei de control a ciclului se mărește cu valoarea rației  $M3$  după fiecare execuție a liniei finale;

— dacă rezultatul operației de decizie asupra propoziției „ $I \leq M2$ ” este „adevărat” se va executa din nou secvența instrucțiunii DO, în caz contrar ciclul s-a încheiat, urmînd a se executa prima instrucțiune executabilă ce urmează instrucțiunii terminale.

Schema de funcționare a unei instrucțiuni DO este dată în figura V.5.

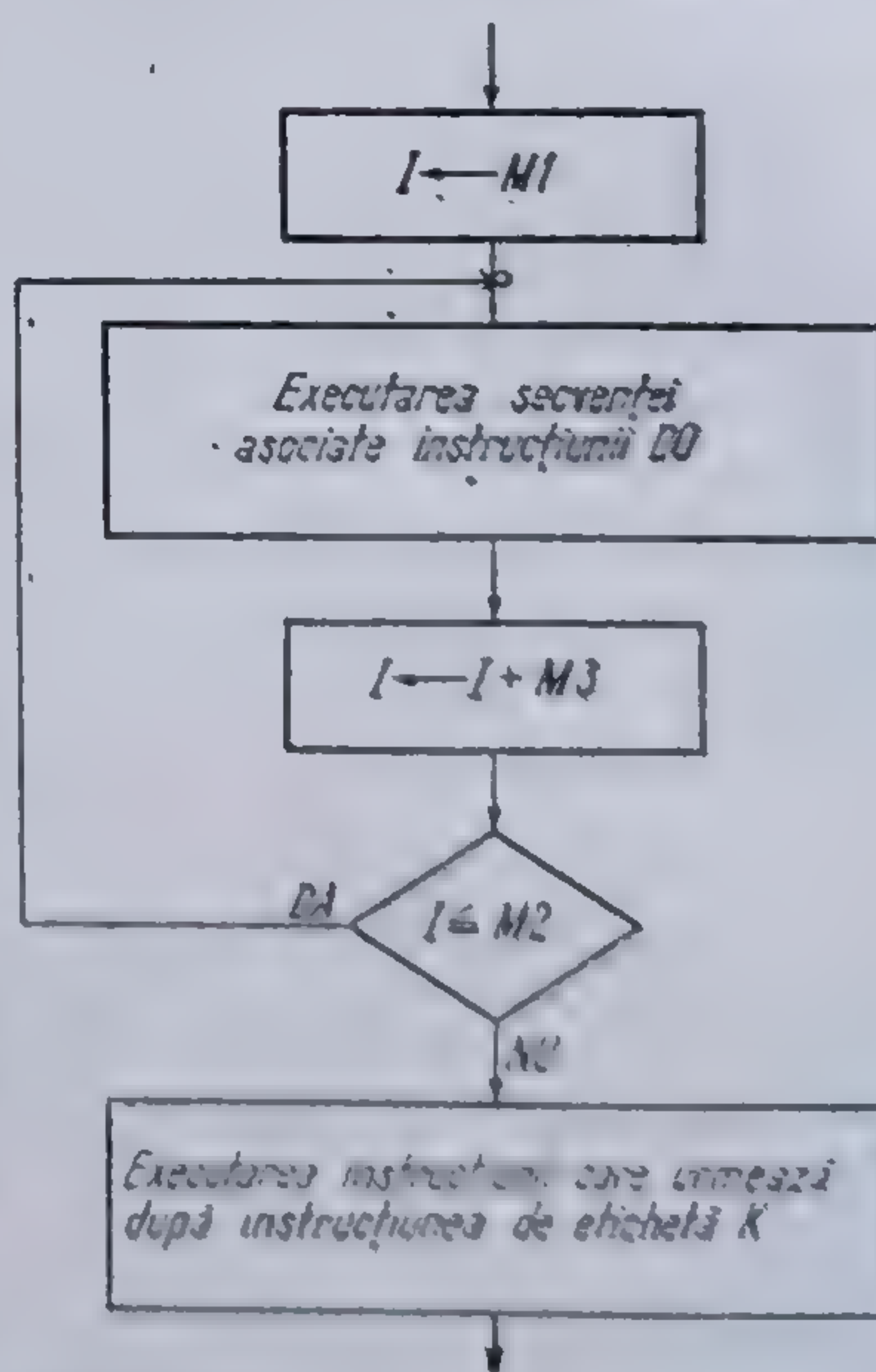


Fig. V.5.



## Observație

Operațiile marcate în blocurile 1, 3 și 4 se execută prin instrucțiunea DO, al cărei format general conține informațiile necesare execuției acestora. Cu alte cuvinte, instrucțiunile corespunzătoare acestor operații nu trebuie să apară distinct în textul programului sursă.

*Intrarea, ieșirea și revenirea în ciclu.*

*Intrarea* într-un ciclu se realizează prin instrucțiunea DO. În secvența ciclului este interzisă modificarea valorilor variabilelor I, M1, M2, M3.

*Ieșirea* din ciclu poate fi:

— *ieșire normală*, adică după epuizarea valorilor variabilei care controlează ciclul se va executa instrucțiunea imediat următoare instrucțiunii terminale a secvenței ciclului;

— *ieșire forțată*, care are loc printr-o instrucțiune de salt din interiorul spre exteriorul secvenței ciclului, înainte de epuizarea tuturor valorilor variabilei de ciclare. În acest caz valoarea variabilei care controlează ciclul este aceea pentru care a avut loc ieșirea forțată din ciclu, putând fi utilizată în continuare în program, dacă este cazul.

Revenirea într-un ciclu, datorită unei ramificări din exteriorul secvenței ciclului spre interiorul acesteia, este permisă numai dacă, în prealabil, a avut loc o ramificare în sens invers, din interiorul secvenței spre exterior și dacă parametrii I, M1, M2, și M3 nu au fost modificați în exteriorul secvenței asociate instrucțiunii DO respective.

## 5. STRUCTURI REPETITIVE DE TIP DO-FOR INCLUSE

În interiorul secvenței unei instrucțiuni DO pot exista și alte instrucțiuni DO. În această situație se spune că există mai multe *cicluri incluse unul în altul*. Pentru ca ciclurile să fie executate, trebuie ca instrucțiunea terminală a ciclului interior să fie scrisă și executată înaintea instrucțiunii terminale a ciclului exterior, sau ca ambele cicluri, interior și exterior, să aibă aceeași instrucțiune terminală.

Se pot pune astfel în evidență următoarele scheme;

	DO $e_1$ I=M1, M2, M3	DO $e_2$ J=N1, N2, N3
	.....	
$e_2$	.....	CONTINUE
$e_1$	CONTINUE	
sau		
	DO $e_1$ I=M1, M2, M3	DO $e_2$ J=N1, N2, N3
	.....	
$e_2$	.....	CONTINUE
$e_1$	CONTINUE	

În cazul a două cicluri incluse unul în altul, o execuție a secvenței ciclului exterior presupune execuția integrală, pentru toate valorile variabilei de control, a ciclului interior.



## 6. CITIREA ȘI SCRIEREA TABLOURILOR

Citirea și scrierea tablourilor se poate realiza în două moduri.

— Se scrie în lista de variabile a instrucțiunilor de intrare/ieșire numai numele tabloului, în acest caz, pentru un tablou bidimensional, citirea/scrierea se execută pe coloane.

*Exemplu:*

```
.....
DIMENSION A(3, 4)
READ (105,2)A
2  FORMAT (12F6.2)
.....
```

Prin execuția acestei secvențe de instrucțiuni, elementele tabloului A(3,4) vor fi depuse în memorie unul după altul, și deci trebuie scrise pe cartelă, în ordinea:

A(1, 1)	A(2, 1)	A(3, 1)	A(1, 2)	A(2, 2)	A(3, 2)	A(1, 3)
A(2, 3)	A(3, 3)	A(1, 4)	A(2, 4)	A(3, 4)		

— Se scrie în lista de variabile a instrucțiunilor de intrare/ieșire numele tabloului urmat de indicii respectivi și valorile lor inițiale, finale și de avans, astfel:

$$\left\{ \begin{array}{l} \text{READ} \\ \text{WRITE} \end{array} \right\} (n, f) ((A(I, J), J = N1, N2, N3), I = M1, M2, M3)$$

sau

$$\left\{ \begin{array}{l} \text{READ} \\ \text{WRITE} \end{array} \right\} (n, f) ((A(I, J), I = M1, M2, M3), J = N1, N2, N3)$$

În primul caz citirea/scrierea tabloului bidimensional A se face pe linii, iar în al doilea caz, pe coloane.

*Exemple:*

a)

```
.....
DIMENSION A(3, 4)
READ (105,2) ((A(I, J), J = 1,4), I=1,3)
2  FORMAT (12F6.2)
.....
```

Prin execuția acestei secvențe de instrucțiuni, elementele tabloului A(3,4) vor fi depuse în memorie unul după altul și deci trebuie scrise pe cartelă, în ordinea:

A(1,1)	A(1,2)	A(1,3)	A(1,4)	A(2,1)	A(2,2)	A(2,3)
A(2,4)	A(3,1)	A(3,2)	A(3,3)	A(3,4)		

b)

```
.....
DIMENSION A(3,4)
.....
WRITE (108,6) ((A(I, J), J = 1,4), I=1,3)
6  FORMAT ('', 4F6.2)
.....
```



Prin execuția acestei secvențe de instrucțiuni vor fi scrise, pe 3 rânduri ale hîrtiei de la imprimantă, următoarele elemente ale tabloului A(3,4):

A(1,1)	A(1,2)	A(1,3)	A(1,4)
A(2,1)	A(2,2)	A(2,3)	A(2,4)
A(3,1)	A(3,2)	A(3,3)	A(3,4)

Programul V.4. Se dă un tablou bidimensional cu N linii și N coloane. Să se scrie programul FORTRAN pentru calculul sumelor elementelor aflate pe diagonala principală, deasupra diagonalei principale și sub diagonală.

### INTREBĂRI ȘI PROBLEME

1. Ce tipuri de structuri repetitive cunoașteți și prin ce se caracterizează fiecare tip?
2. De ce se spune despre unele structuri repetitive că se simulează în FORTRAN?
3. Care sînt instrucțiunile care permit simularea unor tipuri de structuri repetitive?
4. Ce instrucțiune specifică unui anumit tip de structură repetitivă cunoașteți?
5. Să se scrie programul FORTRAN pentru ordonarea descrescătoare a patru numere reale.
6. Să se scrie programul FORTRAN pentru calculul sumelor:

$$S = 1^2 + 3^2 + 5^2 + \dots + N^2$$

$$S = 1 + 1.2 + 1.2.3 + \dots + 1.2.3. \dots .N$$

$$S = 1 + \frac{1}{1.2} + \frac{2}{1.2.3} + \dots + \frac{N-1}{1.2.3. \dots .N}$$

7. Să se scrie programul FORTRAN pentru transformarea unui număr natural N, într-un număr în baza 2.
8. Se dă un tablou unidimensional (vector) format din N numere reale. Să se scrie programul FORTRAN pentru aflarea celui mai mare număr dintre numerele strict negative, precum și numărul său de ordine.
9. Se dau tablourile unidimensionale:

$$X(x_1, x_2, x_3, \dots, x_n)$$

$$Y(y_1, y_2, y_3, \dots, y_n)$$

ale căror elemente sînt numere reale. Se scrie programul FORTRAN pentru calculul expresiei:

$$E = \frac{\sum_{i=1}^n \frac{x_i + y_i}{x}}{\max_{1 \leq i \leq n} \{x_i\}}$$

10. Fiind date tablourile unidimensionale de la problema precedentă să se scrie programul FORTRAN pentru calculul expresiei:

$$E = \frac{\sum_{i=1}^n \left( \frac{1}{x_i} + \frac{1}{y_i} \right)}{\min_{1 \leq i \leq n} \{y_i\}}$$

11. Fiind dat un tablou bidimensional A cu N linii și M coloane, ale cărui elemente sînt numere reale, să se scrie programul FORTRAN pentru aflarea celui mai mare număr și celui mai mic număr din tablou.
12. Fiind dat un tablou bidimensional A cu N linii și M coloane ale cărui elemente sînt numere reale, să se scrie programul FORTRAN pentru determinarea unui tablou unidimensional ale cărui elemente sînt sumele elementelor de pe fiecare coloană a tabloului bidimensional A.



JOB PR54,AN:1234,PN:M1H41  
 COMPILE FORTRAN

C  
 C  
 C  
 C  
 C  
 C  
 C  
 C  
 C

```
*****
*  CALCULUL SUMELOR ELEMENTELOR  *
*  SITUATE PE DIAGONALA PRINCI- *
*  PALA, SUB DIAGONALA PRINCIPALA *
*  SI DEASUPRA DIAGONALEI UNEI  *
*  MATRICE CU N LINII SI M COLOA- *
*  NE.                             *
*****
```

```
INTEGER S1,S2,S3,A(10,10)
READ(105,1) N,(( A( I,J),J=1,N),I=1,N)
1 FORMAT(I2/(10I4))
```

```
S1=0
S2=0
S3=0
```

C

```
*****
```

```
DO 2 I=1,N
  DO 6 J=1,N
    IF (I.LT.J) GO TO 3
    IF (I.GT.J) GO TO 4
    S2=S2+A(I,J)
    GO TO 6
4    CONTINUE
    S3=S3+A(I,J)
    GO TO 6
```

```
3    CONTINUE
    S1=S1+A(I,J)
```

```
6    CONTINUE
```

```
2 CONTINUE
```

C

```
*****
```

```
WRITE 108,5) S1,S2,S3
```

```
5 FORMAT(' ',5X,'S1=',I6,'S2=',I6,'S3=',I
STOP
END
```

LINK  
 RUN

5

```
125-210 12      125 -40 -21  12  11  10
      -14 15      115  20 -40  11  10  9
140 -24 16  17  18
      EOJ
```

PROGRAMUL V.4



13. Fiind dat tabloul bidimensional de la problema precedentă să se scrie programul FORTRAN pentru calculul numărului de elemente strict pozitive, strict negative și nule.
14. Fiind date  $n^2$  numere reale, dispuse într-un tablou bidimensional cu  $N$  linii și  $N$  coloane, să se scrie programul FORTRAN pentru calculul sumelor numerelor dispuse pe pătrate concentrice.
15. Fiind dat un tablou bidimensional cu  $N$  linii și  $N$  coloane să se scrie programul FORTRAN pentru aflarea celui mai mare număr dintre elementele situate deasupra diagonalei principale. Se va scrie apoi programul FORTRAN pentru aflarea celui mai mare număr dintre elementele situate sub diagonala principală.
16. Se dau două tablouri bidimensionale cu  $N$  linii și  $M$  coloane:

$$A = \|a_{ij}\| \quad i = 1, 2, 3, \dots, n; \quad j = 1, 2, 3, \dots, m$$

$$B = \|b_{ij}\| \quad i = 1, 2, 3, \dots, n; \quad j = 1, 2, 3, \dots, m$$

Să se scrie programul FORTRAN pentru determinarea tabloului bidimensional;

$$C = \|c_{ij}\| \quad \text{unde } c_{ij} = \max \{a_{ij}; b_{ij}\} \\ i=1, 2, 3, \dots, n; \quad j=1, 2, 3, \dots, m$$

17. Se dă un tablou unidimensional cu  $N$  componente numere reale. Să se scrie programul FORTRAN pentru înlocuirea fiecărui element cu media aritmetică a celorlalte  $N - 1$  elemente, obținându-se astfel un nou tablou unidimensional.
18. Se dă un tablou bidimensional cu  $N$  linii și  $M$  coloane. Să se scrie programul FORTRAN pentru determinarea unui nou tablou bidimensional în care prima linie devine ultima, a doua devine penultima ș.a.m.d. Aceeași problemă și pentru coloane.
19. Se dă un tablou unidimensional cu  $N$  elemente numere reale. Să se scrie programul FORTRAN prin care elementele extreme (primul și ultimul), cât și cele egal depărtate de elementele extreme să se schimbe între ele.
20. Se dă un tablou unidimensional cu  $N$  elemente numere reale. Să se scrie programul FORTRAN prin care eventualele componente nule să fie trecute la urmă.
21. Să se scrie programul FORTRAN pentru calculul numărului de zile cuprins între două date calendaristice. Datele sînt de forma: zi/lună/an.
22. Se cunoaște data calendaristică sub forma: zi/lună/an. Să se scrie programul FORTRAN pentru a afla ce zi din săptămînă corespunde acestei date.
23. Să se scrie programul FORTRAN pentru aflarea datei calendaristice cînd se cunoaște o dată de referință și numărul de zile ce o desparte de data de referință.
24. Să se scrie programul FORTRAN pentru calculul numărului de zile lucrătoare cuprinse între două date calendaristice.



## CAPITOLUL VI

### PROCEDURI

Algoritmul de rezolvare a unei anumite probleme poate să conțină un grup de operații care se repetă în aceeași ordine în diferite locuri din algoritm.

Programul sursă FORTRAN rezultat în urma codificării algoritmului prezintă astfel aceleași instrucțiuni, scrise în aceeași ordine, în diverse locuri ale programului.

De exemplu, algoritmul de rezolvare a unei probleme necesită extragerea rădăcinii pătrate din trei numere reale pozitive, fiecare radical fiind necesar să se calculeze într-un anumit loc din algoritm.

Programul FORTRAN scris pentru rezolvarea acestei probleme va trebui să conțină de trei ori, în trei locuri distincte, instrucțiunile pentru calculul rădăcinii pătrate dintr-un număr real pozitiv.

De asemenea, programe FORTRAN diferite pot conține un ansamblu de instrucțiuni comun.

De exemplu, rezolvarea mai multor probleme presupune aflarea soluției unui anumit sistem liniar de  $n$  ecuații cu  $n$  necunoscute.

Aceasta înseamnă că programele sursă FORTRAN scrise pentru problemele respective, vor conține o parte comună, determinată de ansamblul instrucțiunilor necesare pentru rezolvarea unui sistem liniar de  $n$  ecuații cu  $n$  necunoscute.

Se pune astfel problema simplificării muncii de programare, în sensul creării și utilizării unor *proceduri* care, prin diverse tehnici de programare, să fie apelate și executate în cadrul unui program ori de câte ori acesta o cere.

În cadrul exemplelor date anterior procedurile vor fi utilizate pentru calculul radicalului de ordinul 2, respectiv rezolvarea unui sistem liniar de  $n$  ecuații cu  $n$  necunoscute.

În limbajul FORTRAN există următoarele *tipuri de proceduri*: *funcții* și *subprograme*.

*Funcțiile* sînt proceduri pentru calculul unei singure valori, care va fi folosită în continuare în programul care a utilizat funcția respectivă.

*Subprogramele* sînt proceduri care permit calculul unui număr oarecare de valori necesare programului care a utilizat subprogramul respectiv. Subprogramele pot fi utilizate nu numai pentru calculul unui anumit număr de valori, dar și pentru executarea anumitor operații necesare rulării programului pe calculator (operații de intrare/ieșire etc.).

Programul care apelează o anumită procedură se va numi *program apelant*. Procedura care este apelată de un program apelant se va numi *program apelat*. Orice program apelat poate deveni program apelant dacă va apela o procedură, alta decît cea care apelează programul respectiv.

Un program apelant care nu este, la rîndul său, apelat de altul, se numește *program principal*.

Programul principal și procedurile sale (apelate și apelante) constituie mai multe unități de program ale aceluiași program.



## 1. FUNCȚII

Se disting două categorii de funcții:

- funcții-formulă sau funcții definite;
- funcții standard.

a. **Funcția-formulă** este definită de expresia care servește la calculul valorii funcției.

Funcția-formulă este o instrucțiune neexecutabilă și neetichetabilă, care se scrie în program înaintea tuturor instrucțiunilor executabile, după instrucțiunile de declarare a tipului și dimensiunilor.

Format general

$$F(P_1, P_2, P_3, \dots, P_N) = E$$

unde:

—  $F$  este numele funcției. Acest nume poate fi obiectul unei declarații explicite de tip, altfel tipul lui  $F$  este definit de regula implicită.

—  $P_1, P_2, P_3, \dots, P_N$  parametri (argumente) formali (fictivi, convenționali), fiind nume de variabile simple. Numele parametrilor formali nu servesc decât la a indica tipul, numărul și ordinea argumentelor; acești parametri putând avea nume identice cu nume simbolice folosite undeva în aceeași unitate de program. Tipul parametrilor este determinat implicit prin prima literă a numelui lor, sau explicit printr-o declarație de tip.

—  $E$  este numele unei expresii formată din parametri formali, constante, funcții standard sau funcții anterior definite. Expresia  $E$  este o expresie aritmetică dacă funcția este de tip real, complex, dublă precizie, sau o expresie logică dacă funcția este de tip logic.

Expresia  $E$  se evaluează atunci când în program, într-o expresie apare numele funcției urmat de parametri actuali (efectivi, reali), separați prin virgulă și închiși între paranteze. Parametrii actuali trebuie să coincidă ca număr, ordine și tip cu parametri formali corespunzători. Parametrii actuali pot fi:

- constante,
- nume de variabile,
- expresii de același tip cu parametrul formal corespunzător.

b. **Funcțiile standard** definesc proceduri (subprograme) scrise de firmele constructoare de calculatoare, depuse în biblioteca de subprograme standard (BSS) a sistemului.

Pentru calculul valorii unei astfel de funcții, programatorul nu trebuie să scrie procedura respectivă, ci doar să apeleze subprogramul corespunzător din BSS.

Numele acestor funcții sînt cuvinte rezervate. Fiecare funcție se apelează într-o expresie în care figurează numele funcției, urmat de parametri efectivi separați prin virgulă și incluși între paranteze.

Funcțiile standard aflate în biblioteca de subprograme standard a sistemului FELIX C-256 sînt prezentate în tabelul VI.1.

Tipul funcției și al parametrilor este:

- $C$  complex
- $D$  dublă precizie
- $R$  real
- $I$  întreg



TABELUL VI.1

Funcția		Parametri		Definiția
Nume	Tip	Tip	Număr	
ABS IABS DABS CABS	R I D R	R I D C	1 1 1 1	Valoare absolută
AINT INT IDINT	R I I	R R D	1 1 1	Valoarea parametrului se rotunjește la numărul întreg imediat inferior în valoare absolută
AMOD MOD	R I	R I	2 2	Restul împărțirii întregi a valorii primului parametru la al doilea
AMAXO AMAXI MAXO MAXI DMAXI	R R I I D	I R I R D	$\geq 2$ $\geq 2$ $\geq 2$ $\geq 2$ $\geq 2$	Alege valoarea cea mai mare dintre valorile parametrilor: $\max(a_1, a_2, \dots, a_n)$
AMINO AMINI MINO MINI DMINI	R R I I D	I R I R D	$\geq 2$ $\geq 2$ $\geq 2$ $\geq 2$ $\geq 2$	Alege valoarea cea mai mică dintre valorile parametrilor: $\min(a_1, a_2, \dots, a_n)$
FLOAT	R	I	1	Conversia unui întreg în real
DFLOAT	D	I	1	Conversia unui întreg în dublă precizie
IFIX	I	R	1	Conversia unui real în întreg
DIM IDIM	R I	R I	2 2	Calculează diferența dintre valoarea primului parametru și minimul valorilor primului și celui de-al doilea parametru
SIGN ISIGN DSIGN	R I D	R I D	2 2 2	Valorii absolute a primului parametru i se atribuie semnul valorii celui de-al doilea parametru
REAL	R	C	1	Partea reală a valorii unui parametru complex
AIMAG	R	C	1	Partea imaginară a valorii unui parametru complex
DBLE	D	R	1	Parametru real convertit în dublă precizie
CONJG	C	C	1	Conjugatul unui parametru complex
EXP DEXP CEXP	R D C	R D C	1 1 1	$e^a$
ALOG DLOG CLOG	R D C	R D C	1 1 1	$\log_e(a)$
ALOG10 DLOG10	R D	R D	1 1	$\log_{10}(a)$
SIN DSIN CSIN	R D C	R D C	1 1 1	$\sin(a)$
COS DCOS CCOS	R D C	R D C	1 1 1	$\cos(a)$
SQRT DSQRT CSQRT	R D C	R D C	1 1 1	Extragerea rădăcinii pătrate
ATAN DATAN	R D	R D	1 1	$\text{Arc tg}(a)$



```

•      JOB PR61,AN:1234,PN:MIHAI
•      COMPILE FORTRAN
C
C      *****
C      * UTILIZAREA UNEI FUNCTII DEFINITE *
C      *****
C
      F(X)=(X**2+(7+X**2)**0.5)/(1+X**2)**0.5
      READ(105,1) N,A,B
1  FORMAT(I4,2F5.2)
      S=1./2*(F(A)+F(B))
      M=N-1
      DO 2 K=1,M
          X=K*(B-A)/N
          S=S+F(A+X)
2  CONTINUE
      S=S*(B-A)/N
      WRITE(108,3) S
3  FORMAT(' ',10X,'S=',F14.6)
      STOP
      END
•      LINK
•      RUN
•      100  0      4
•      EOJ

```

#### PROGRAMUL VI.1

Programul VI.1. Să se calculeze:

$$S = \frac{B-A}{N} \left[ \frac{1}{2} F(A) + \sum_{K=1}^{N-1} F\left(A + K \frac{B-A}{N}\right) + \frac{1}{2} F(B) \right]$$

știind că:

$$N = 100, A = 0, B = 4.$$

$$F(X) = \frac{X^2 + \sqrt{7 + X^2}}{\sqrt{1 + X^2}}$$

## 2. SUBPROGRAME

Se disting, în limbajul FORTRAN, două tipuri de subprograme:

- subprograme de tip FUNCTION;
- subprograme de tip SUBROUTINE.

a. Subprogramul de tip FUNCTION constituie o unitate de program separată de programul care îl apelează.

Un astfel de subprogram este compilat separat de programul apelant.



Un subprogram de tip FUNCTION este format din următoarele elemente:

- instrucțiunea de definiție FUNCTION;
- corpul subprogramului;
- instrucțiunea END.

Instrucțiunea FUNCTION este neexecutabilă și neetichetabilă. Formatul general al instrucțiunii este:

$t$  FUNCTION F(P1, P2, P3, ..., PN)

unde:

- $t$  reprezintă tipul funcției, care poate fi:

INTEGER  
REAL  
DOUBLE PRECISION  
COMPLEX  
LOGICAL

Tipul funcției poate să nu figureze explicit în formatul instrucțiunii dacă este definit implicit prin numele funcției (F). În cazul în care tipul funcției figurează explicit, în programul apelant trebuie să figureze o declarație explicită asupra tipului funcției.

- F este numele simbolic al funcției (identificator de variabilă).

- P1, P2, P3, ..., PN sînt *parametri formali*. Aceștia pot fi: nume de variabile, nume de tablou, nume de subprograme deja definite. Numărul parametrilor formali poate fi oarecare, dar trebuie să existe cel puțin unul. Numele parametrilor formali nu pot să apară în instrucțiunile DATA, COMMON, EQUIVALENCE.

Corpul subprogramului de tip FUNCTION conține o serie de instrucțiuni printre care, obligatoriu, o instrucțiune de atribuire la care, în partea stîngă figurează o variabilă cu numele subprogramului FUNCTION și cel puțin o instrucțiune RETURN. Instrucțiunea RETURN asigură revenirea în programul apelant. Formatul instrucțiunii RETURN este:

RETURN

De asemenea, corpul subprogramului, în afară de instrucțiunile de declarare a tipului și dimensiunilor pentru variabile cu care se lucrează numai în subprogram, trebuie să conțină și toate instrucțiunile de declarare a tipului și dimensiunile care figurează în programul apelant și care se referă la variabilele comune programelor apelant și apelat. Corpul subprogramului nu trebuie să conțină instrucțiunile:

SUBROUTINE  
BLOCK DATA<sup>1</sup>  
FUNCTION

Subprogramul de tip FUNCTION se execută atunci cînd, în programul apelant se întâlnește într-o expresie numele funcției urmat de *parametrii efectivi*, care trebuie să coincidă ca număr, ordine și tip, cu parametrii formali corespunzători. Parametrii efectivi pot fi: constante, nume de variabile, nume de variabile indexate, nume de tablouri, nume de subprograme, expresii de același tip cu parametrul formal corespunzător.

Parametrii efectivi se împart în două categorii:

- *parametri de intrare*, ale căror valori constituie datele de intrare necesare obținerii rezultatelor dorite;

- *parametri de ieșire*, cărora li se atribuie valorile calculate în subprogram.



Subprogramul de tip FUNCTION întoarce în programul apelant atât valoarea atribuită variabilei cu numele funcției, cât și, dacă este cazul, valorile atribuite parametrilor de ieșire.

Programul VI.2. Să se calculeze:

$$C_N^K = \frac{N!}{K!(N-K)!}$$

folosind o procedură de tip FUNCTION pentru calculul factorialului.

```

•      JOB PR62,AN:1234,PN:MIHAI
•      COMPILE FORTRAN
C
C      *****
C      * UTILIZAREA UNEI PROCEDURI DE TIP *
C      * FUNCTION PENTRU CALCULUL FACTO- *
C      * RIALULUI UNUI NUMAR              *
C      *****
C
      READ (105,1) N,K
1  FORMAT(2I2)
      C=FACT(N)/FACT(K)*FACT(N-K)
      WRITE 108,2) N,K,C
2  FORMAT(' ', 'COMBINARI DE ',I2,' CITE ',I2,
*      ' ',F5.0)
      STOP
      END
C      PROCEDURA TIP FUNCTION
      FUNCTION FACT(N)
      FACT=1
      DO 3 K=1,N
          FACT=FACT*K
3  CONTINUE
      RETURN
      END
•      LINK
•      RUN
10 6
•      EOJ

```

PROGRAMUL VI.2

b. Subprogramul de tip SUBROUTINE constituie o unitate separată de program, formată dintr-o succesiune de instrucțiuni care se compilează separat de programul apelant.

Ca și la subprogramul de tip FUNCTION, se disting următoarele elemente ale unui subprogram de tip SUBROUTINE:

- instrucțiunea de definiție SUBROUTINE
- corpul subprogramului;
- instrucțiunea END.



Instrucțiunea SUBROUTINE este neexecutabilă și netichetabilă. Formatul general al instrucțiunii este:

SUBROUTINE S(P1, P2, P3, ..., PN)

unde:

- S este numele simbolic al subprogramului;
- P1, P2, P3, ..., PN sînt *parametri formali* care pot fi: nume de variabile, nume de tablouri, nume de subprograme, asterisc. Numărul parametrilor formali poate fi oarecare, totuși este posibil să nu existe nici un parametru formal. În lista parametrilor formali se deosebesc două tipuri:

- *parametrii de intrare*, ale căror valori sînt preluate, de către subprogram, din programul apelant;

- *parametrii de ieșire*, ale căror valori sînt calculate în subprogram, urmînd a fi transmise programului apelant.

Corpul subprogramului poate conține orice instrucțiune cu excepția:

- FUNCTION

- BLOCK DATA

- o altă instrucțiune SUBROUTINE.

Reîntoarcerea rezultatelor în programul apelant este asigurată prin instrucțiunea RETURN și are loc la instrucțiunea imediat următoare (din programul apelant) instrucțiunii de apelare a subprogramului.

Instrucțiunea de apelare a subprogramului de tip SUBROUTINE este instrucțiunea CALL.

*Format general*

CALL S(A1, A2, A3, ..., AN)

unde:

- S este numele subprogramului (același din instrucțiunea SUBROUTINE)

- A1, A2, A3, ..., AN sînt *parametrii efectivi* care trebuie să coincidă ca număr, ordine și tip cu parametrii formali corespunzători din instrucțiunea SUBROUTINE. Parametrii efectivi pot fi:

- constante,

- nume de variabile,

- nume de variabile indexate,

- nume de tablou,

- nume de subprograme,

- expresii de același tip cu parametrul formal corespunzător,

- etichete precedate de ampersant (&).

● *Ieșiri multiple din subprogramul de tip SUBROUTINE.* Există posibilitatea reîntoarcerii din subprogram în programul apelant și în alte puncte decît la instrucțiunea imediat următoare instrucțiunii CALL.

În acest caz, printre argumentele efective ale instrucțiunii CALL trebuie să se afle și etichetele instrucțiunilor la care se face reîntoarcerea, precedate de caracterul &. Acestor etichete le corespund ca argumente formale în instrucțiunea SUBROUTINE un număr egal de asteriscuri.

Reîntoarcerea la fiecare din instrucțiunile ale căror etichete sînt argumente în instrucțiunea CALL se asigură printr-o instrucțiune RETURN i scrisă în subprogram. Formatul instrucțiunii RETURN i este:

RETURN i



unde  $i$  este o constantă întreagă pozitivă, cel mult egală cu numărul asteriscurilor din lista argumentelor formale ale instrucțiunii SUBROUTINE. Când  $i$  are valoarea  $k$ , reîntoarcerea în programul apelant are loc la instrucțiunea a cărei etichetă are numărul de ordine  $k$  în lista etichetelor din instrucțiunea CALL.

### Exemplu:

Să se calculeze într-un subprogram de tip SUBROUTINE suma elementelor din fiecare coloană a tabloului A (10, 20). Citirea elementelor tabloului A și afișarea rezultatelor se execută în programul apelant.

```

C      PROGRAM APELANT
        DIMENSION A(10, 20), V(20)
        READ (105, 1) ((A(I, J), J = 1, 20), I = 1, 10)
1      FORMAT (20F4.1)
        CALL SUMA (A, V)
        WRITE (108, 2) (V (J), J = 1, 20)
2      FORMAT ('/' '\ 'SUMELE DE PE COLOANE)
        A SINT '/' '\ 20F6.1)
        STOP
        END
C      SUBPROGRAM TIP SUBROUTINE
        SUBROUTINE SUMA (AA, VV)
        DIMENSION AA(10, 20), VV(20)
        DO 3 J = 1, 20
            VV(J) = 0
            DO 4 I = 1, 10
                VV(J) = VV(J) + AA(I, J)
4          CONTINUE
3        CONTINUE
        RETURN
        END

```

În acest program parametrii formali sînt AA și VV, iar parametrii efectivi sînt A și V.

*Programul VI.3.* Să se rezolve ecuația  $ax^2 + bx + c = 0$ , utilizînd un subprogram de tip SUBROUTINE pentru calculul discriminantului și a rădăcinilor ecuației.

În subprogramul DISCR se calculează discriminantul ecuației de gradul II și în funcție de valoarea pozitivă, nulă sau negativă a acestuia se calculează valorile rădăcinilor. Prin instrucțiunea RETURN  $i$  ( $i = 1, 2, 3$ ) se revine în programul principal la una din instrucțiunile de scriere cu etichetele 20, 30 sau 40, etichete care, precedate de ampersant, figurează ca argumente în lista instrucțiunii de apelare CALL.

### Observație

Datorită reîntoarcerii din subprogramul de tip SUBROUTINE în mai multe puncte ale programului apelant, nu se respectă principiile programării structurale care, după cum se știe, cer o singură ieșire și o singură intrare din și respectiv în programul apelant. Din acest motiv instrucțiunea RETURN  $i$  nu se utilizează în programarea structurată.



### 3. INSTRUCȚIUNEA EXTERNAL

Se utilizează în cazul în care numele unui subprogram figurează ca argument în alt subprogram (de tip FUNCTION sau SUBROUTINE).

Instrucțiunea este neexecutabilă și neetichetabilă și se scrie în programul apelant înaintea tuturor instrucțiunilor executabile.

Formatul general al instrucțiunii EXTERNAL este:

EXTERNAL,  $nume_1, nume_2, \dots, nume_n$

unde:

$nume_1, nume_2, \dots, nume_n$  sînt numele unor subprograme care apar ca argumente efective în instrucțiunile de apelare a altor subprograme.

În lista de argumente a instrucțiunii EXTERNAL, pot apare atît numele unor subprograme scrise de utilizator, cît și numele unor funcții standard.

#### Observație

Instrucțiunea EXTERNAL nu poate apare într-un subprogram.

#### Exemplu:

```
C      PROGRAM APELANT
      .....
      EXTERNAL SIN, DELTA
      .....
      CALL ETTA (A, B, DELTA, C)
      .....
      CALL ETTA (A1, B1, SIN, C1)
      .....
      STOP
      END
C      SUBPROGRAM DE TIP SUBROUTINE
      SUBROUTINE ETTA (U, V, W, T)
      .....
      U = W(V) + T
      .....
      RETURN
      END
C      SUBPROGRAM DE TIP FUNCTION
      FUNCTION DELTA (X)
      .....
      DELTA = X**2 + 2
      RETURN
      END
```

În programul apelant s-au declarat, în instrucțiunea EXTERNAL, numele subprogramului DELTA de tip FUNCTION, scris de programator și numele funcției standard SIN.

La întîlnirea instrucțiunii

CALL ETTA (A, B, DELTA, C)

se execută programul de tip SUBROUTINE ETTA (U, V, W, T) cu parametrii efectivi A, B, DELTA, C atribuiți, respectiv parametrilor U, V, W, T.



În subprogramul ETTA se apelează subprogramul de tip FUNCTION  
FUNCTION DELTA (X)  
prin instrucțiunea

$$U = W(V) + T$$

La întâlnirea acestei instrucțiuni se trece la executarea subprogramului  
FUNCTION DELTA (X)

cu parametrul formal X corespunzător parametrului formal V și parametrului efectiv B.

Numele funcției

DELTA

corespunde parametrului formal W din subprogram, iar în programul apelant numele se păstrează.

La întâlnirea instrucțiunii

CALL ETTA (A1, SIN, B1, C1)

se execută subprogramul de tip SUBROUTINE

SUBROUTINE ETTA (U, V, W, T)

cu parametrii efectivi A1, SIN, B1, C1 atribuiți respectiv parametrilor U, V, W, T.

În subprogramul ETTA se apelează funcția standard SIN prin instrucțiunea

$$U = W(V) + T$$

Programul VI.4. Să se traseze graficul funcției  $Y(X) = \frac{1}{2} \sin(2X + 1)$  utilizând un subprogram de tip SUBROUTINE.

### Soluție

Pentru scrierea unui program de trasare a graficului unei funcții se poate proceda astfel:

— Se definește un tablou unidimensional (vector) cu 101 componente (numărul de componente poate fi și altul). Două și numai două din aceste componente vor conține caracterul alfanumeric de trasare (asterisc sau orice alt semn), iar celelalte vor cuprinde spații.

Caracterul deplasabil va fi atribuit întotdeauna unei aceleiași componente, pentru a marca axa OX, iar cea de a doua componentă căreia îi este atribuit acest caracter este variabilă, corespunzătoare valorii funcției. Prin imprimarea repetată a vectorului astfel constituit se obține graficul funcției. Rezultă că pentru fiecare imprimare, trebuie determinați indicii componentelor cărora li se atribuie caracterul deplasabil.

— Stabilirea poziției axei OX. Indicele componentei care marchează axa OX este constant.

Fie  $v_2$  valoarea maximă a funcției (pe un anumit interval),  $v_1$  valoarea minimă, iar  $v$  valoarea curentă a funcției.

Dacă  $v_2$  și  $v_1$  au semne contrarii, indicele I al componentei care va reprezenta axa OX este:

$$I_{ox} = \left\lceil \left| \frac{v_1}{v_2 - v_1} \cdot 100 \right| \right\rceil + 1$$





Fig. VI.1.

unde parantezele drepte indică faptul că se extrage partea întreagă a modului rezultatului.

Dacă  $v_1$  și  $v_2$  sînt ambele negative, axa  $OX$  se va desena la dreapta de tot (în dreptul valorii  $v_2$ ), iar dacă  $v_1$  și  $v_2$  sînt ambele pozitive, axa  $OX$  se va desena la stînga de tot (în dreptul valorii  $v$ ).

— *Stabilirea poziției punctului curent.* Indicele  $I$  al componentei care marchează poziția punctului curent este dat de formula:

$$I_e = \left[ \frac{v - v_1}{v_2 - v_1} \cdot 100 \right] + 1$$

În situația în care aprecierea inițială a valorilor extreme nu este corectă se prevede, ca în loc de imprimarea unui punct, să se imprime valoarea efectivă a funcției.

## INTREBĂRI ȘI PROBLEME

1. Ce tipuri de proceduri cunoașteți?
2. Cum se definește o funcție-formulă?
3. Prin ce se caracterizează funcțiile standard?
4. Ce tipuri de subprograme cunoașteți?
5. Să se calculeze și să se afișeze valoarea expresiei

$$E = (ax^2 - bx + c) : (dx^2 - ex + f)$$

utilizînd o funcție-formulă definită astfel:

$$F(y) = ay^2 - by + c$$

6. Se consideră o funcție logică de două variabile

$$F(X, Y) = (\bar{X} \vee Y) \wedge (\bar{Y} \vee X) \vee Y$$



Să se scrie programul FORTRAN prin care să se evalueze funcția pentru cele 4 valori posibile ale argumentelor:

X = .TRUE.	Y = .TRUE.
X = .TRUE.	Y = .FALSE.
X = .FALSE.	Y = .TRUE.
X = .FALSE.	Y = .FALSE.

7. Să se calculeze:

$$S = \frac{1 + 2 + 3 + 4 \dots + N}{(N + 1) + (N + 2) + \dots + L}$$

scriind o procedură de tip FUNCTION pentru calculul sumei de la numărător și numitor.

8. Se dau două mulțimi reale, cu număr finit de elemente. Să se scrie programul FORTRAN pentru determinarea reuniunii celor două mulțimi. Procedura de aflare a mulțimii reuniune se va realiza într-un subprogram de tip SUBROUTINE.

9. Să se scrie programul FORTRAN pentru determinarea intersecției mulțimilor de la problema precedentă. Procedura se va realiza într-un subprogram de tip SUBROUTINE.

10. Să se scrie programul FORTRAN pentru determinarea diferenței a două mulțimi de numere reale, cu număr finit de elemente.

Procedura se va realiza într-un subprogram de tip SUBROUTINE.

11. Se dă un pachet de 6 cartele cu structura:

- în coloanele 1–2 un număr întreg  $N(1 \leq N \leq 4)$
- în coloanele 3–8 un număr real  $A(F6.2)$
- în coloanele 9–14 un număr real  $B(F6.2)$ ;
- în coloanele 15–19 un număr real  $C(F5.3)$ .

Să se scrie o procedură de tip SUBROUTINE pentru calculul funcției:

$$F(N) = \begin{cases} A + B + C & \text{dacă } N = 1 \\ \text{MAX}(A, B, C) & \text{dacă } N = 2 \\ \text{MIN}(A, B, C) & \text{dacă } N = 3 \\ \text{SQRT}(A/B) & \text{dacă } N = 4 \end{cases}$$

Se cere să se tipărească în programul apelant numai valorile pozitive ale funcției ceea ce implică două întoarceri în funcție de valoarea F rezultată.

12. Să se traseze graficul funcției  $y(x) = e^{\sin x}$  utilizând un subprogram de tip SUBROUTINE pentru determinarea punctelor graficului.



## CAPITOLUL VII

### FIȘIERE

Limbajul FORTRAN este destinat problemelor cu caracter științific, care presupun, în general puține date de intrare/ieșire și algoritmi de complexitate mare, ceea ce conduce la programe ample, cu multe variabile și instrucțiuni.

Utilizarea metodelor matematice moderne de calcul în rezolvarea problemelor economice a modificat însă oarecum caracteristicile problemelor rezolvabile cu ajutorul limbajului FORTRAN, în sensul că se păstrează natura complexă a algoritmului, dar volumul datelor de intrare/ieșire crește considerabil.

Prin urmare trebuie evitată depășirea capacității memoriei disponibile. În acest scop, limbajul FORTRAN permite reducerea memoriei interne necesare execuției unui program prin micșorarea atât a numărului de date, cât și a numărului de instrucțiuni aflate simultan în memorie.

Acest lucru este posibil, pe de o parte prin organizarea volumului de informații de intrare/ieșire sub forma unor fișiere, iar pe de altă parte prin folosirea unei tehnici speciale de programare — numită segmentare — care constă în împărțirea programului în unități care pot fi separat încărcate în memorie, în vederea execuției.

#### 1. UTILIZAREA FIȘIERELOR ÎN PROGRAMELE FORTRAN

În general, un fișier se prelucrează în trei moduri:

- creare (scriere de articole în fișier);
- consultare (citirea articolelor din fișier);
- actualizare.

Actualizarea unui fișier constituie o prelucrare complexă care presupune atât citirea cât și scrierea în fișier a unor articole. Modurile uzuale de actualizare sînt:

- modificarea conținutului unor articole;
- ștergerea unor articole din fișier;
- adăugarea unor articole în fișier.

Posibilitățile de realizarea diverselor moduri de actualizare depind de modul de acces la articolele din fișier (acces direct sau secvențial) și de organizarea fișierului.

Limbajul FORTRAN permite prelucrarea fișierelor organizate secvențial și a fișierelor nedefinite. Suportul este de obicei organizat standard.

Fișierele pot fi:

- pe cartele;
- la imprimantă;
- pe benzi magnetice (organizate standard sau nestandard);
- pe discuri magnetice.



Fișierele pe cartele și la imprimantă sînt întotdeauna organizate secvențial.

Organizarea fișierelor pe suport magnetic este nedefinită. Dacă astfel de fișiere urmează însă să fie prelucrate în programe care nu admit decît fișiere organizate secvențial, ele pot fi declarate în organizarea secvențială, în acest caz fiind impuse însă restricții la prelucrare.

Pentru a putea prelucra un fișier printr-un program, sistemul de operare și anume o componentă a sa — Sistemul de Gestiunea Fișierelor (SGF) — trebuie să primească o serie de informații cu caracter general, care să-i permită:

- identificarea suportului pe care este depus fișierul;
- identificarea fișierului pe suport;
- identificarea înregistrării în fișier.

O parte din aceste informații sînt furnizate prin intermediul cartelelor de comandă INIT, ALLOC, ASSIGN și LABEL, iar o altă parte se transmite sistemului prin intermediul programului FORTRAN.

Într-un program FORTRAN aceste informații se referă la:

- organizarea fișierului;
- formatul și dimensiunea articolelor din fișier;
- tipul și modul de organizare a suportului;
- numărul și dimensiunea zonelor tampon folosite;
- identificarea fișierului în programul respectiv.

După cum se știe există anumite fișiere de intrare/ieșire cu regim special:

— fișierul de intrare al sistemului (sau de lucrări), care conține lucrările destinate execuției și

— fișierul de ieșire al sistemului (sau de editare), care conține rezultate legate de lucrări (text sursă, situația în fiecare fază, conținutul memoriei etc.).

Aceste fișiere pot fi pe orice suport. Se consideră în general că fișierul de intrare este pe cartele, iar fișierul de editare pe hîrtia de imprimat sau pe cartele.

Fișierele de intrare/ieșire ale sistemului au indexuri prestabilite, și anume:

- fișier de lucrări: \* 1;
- fișier de editare la imprimantă: \* 2;
- fișier de editare pe cartele perforate: \* 3.

Într-un program FORTRAN referirile la fișierele sistem de intrare/ieșire se fac prin intermediul unor identificatori care se numesc numere FORTRAN-standard.\*

Acestea sînt:

105 pentru fișierul de lucrări;

108 pentru fișierul de editare la imprimantă;

106 pentru fișierul de editare pe cartele perforate.

Datele de intrare ale unui program, cînd nu sînt organizate ca fișiere, sînt întotdeauna cuprinse în fișierul de lucrări. La fel și rezultatele programului fac parte din fișierul sistem de editare. Din acest motiv în instrucțiunile de intrare/ieșire descrise în capitolele precedente apăreau numerele FORTRAN 105 și 108. O instrucțiune de citire era de forma:

READ (105, f) listă;

\* Aceste numere sînt specifice sistemului SIRIS 2/3. Pentru alte sisteme de operare sînt diferite. De exemplu, un program FORTRAN destinat rulării pe un calculator IBM-360, va avea ca numere FORTRAN pentru fișierele sistem 1, 3, respectiv 2.  
Se observă astfel că limbajul FORTRAN nu este complet independent de mașină.



iar o instrucțiune de scriere:

WRITE (108, f) listă.

Implicit se consideră că orice program FORTRAN utilizează fișierele \* 1 și \* 2, compilatorul FORTRAN inserînd automat tabelele de descriere ale acestor fișiere.

Din acest motiv, dacă fișierele utilizator sînt cuprinse în fișierul de lucrări sau în fișierul de editare la imprimantă, ele nu mai necesită nici o descriere specială în cadrul programului FORTRAN.

Folosirea oricăror alte fișiere necesită declararea lor în programul FORTRAN.

## 2. DECLARAREA FIȘIERELOR

Declararea fișierelor în programele FORTRAN se realizează prin intermediul ordinelor \* DEFINE FILE sau \* USED FILE. Între aceste două ordine există o singură deosebire și anume, în cazul folosirii lui \*USED FILE nu se mai implantează în memorie tabela de descriere a fișierului \* 1, deci se presupune că datele de intrare ale programului nu sînt conținute în fișierul de lucrări.

Un ordin de declarare a fișierelor are următorul *format general*:

\* ordin *idex* (*listă de parametri*) = *n*

și conține:

- caracterul \* în colcana 1;
- caracterul blank, — în coloana 2;
- coloanele 3 ÷ 14 DEFINE FILE urmat de un blank, sau coloanele 3 ÷ 12 USED FILE urmat de un blank;
- în continuare, lista de argumente;
- *idex* reprezintă *idexul* de *exploatare* al fișierului și este format dintr-o literă și o cifră. Litera trebuie să fie aceeași cu cea asociată fișierului prin cartela ASSIGN, iar cifra poate fi 1, 2 sau 0, indicînd modul de prelucrare (tabelul VII.1);

TABELUL VII.1

MOD DE PRELUCRARE	CIFRA DIN IDEX
SCRIERE	2
CITIRE	1
SCRIERE/CITIRE	0

— *n* este un număr întreg, pozitiv, diferit de zero și reprezintă identificatorul fișierului în instrucțiunile FORTRAN specifice prelucrării fișierelor. Fișierele distincte sînt identificate prin numere FORTRAN distincte. Dacă din anumite motive, la scrierea programului, același fișier a fost referit prin mai multe numere FORTRAN *n*<sub>1</sub>, *n*<sub>2</sub>, ... *n*<sub>i</sub>, acest lucru se specifică în DEFINE FILE sau USED FILE astfel:

\* ordin *idex* (*lista de argumente*) = *n*<sub>1</sub> = *n*<sub>2</sub> = *n*<sub>i</sub>



Acest lucru permite și prelucrarea programelor FORTRAN scrise pentru alte sisteme de operare care în instrucțiunile de intrare/ieșire folosesc pentru fișierele sistem alte numere FORTRAN decât cele ale sistemului SIRIS. Astfel, dacă într-un program FORTRAN fișierele sistem \*1, \*2 sînt folosite cu numerele 1, respectiv 3, pentru a putea fi executat programul trebuie să conțină instrucțiunea:

\* DEFINE FILE \*1 = 1,\*2 = 3

prin care fișierului \*1 i se asociază, pe lângă 105 și numărul 1, iar fișierul \*2 poate fi identificat prin numerele 108 și 3. În acest exemplu nu apare nici o listă de argumente, orice descriere suplimentară fiind inutilă, deoarece se referă la fișierele sistem.

#### Observație

Descrierile explicite sînt prioritare față de cele implicite. Astfel, dacă într-un program apare:

\* DEFINE FILE \*1 = 108,\*2 = 105

fișierul de lucrări va avea asociat numai numărul FORTRAN 108, iar fișierul de editare la imprimantă numai numărul 105;

— *lista de parametri* conține informațiile legate de tipul și organizarea suportului, structura și organizarea fișierului, numărul și dimensiunea zonelor tampon.

Parametrii care pot figura într-o astfel de listă sînt descriși în tabelul VII.2.

TABELUL VII.2

Parametrul	Descrierea parametrului
DVT: <i>dd</i> tipul dispozitivului periferic	<div> <div><i>dd</i> poate fi:</div> <div> <div>CD disc MD200</div> <div>AD disc MD50</div> <div>BD disc MD100</div> <div>CR cititor cartele</div> <div>CP perforator cartele</div> <div>PR imprimantă</div> </div> </div> <div> <div>MT bandă magnetică</div> <div>RD disc DIAM</div> <div>FD disc RAD</div> <div>DK disc oarecare</div> <div>MD disc MD25</div> </div>
[RCF: <i>f</i> ]  formatul articolului	<div> <div><i>f</i> poate fi:</div> <div> <div>D fișier pe disc în acces direct</div> <div>B fișier exploatat cu BUFFER IN/OUT</div> <div>U format nedefinit</div> <div>F format fix</div> <div>FB format fix blocat</div> <div>V format variabil</div> <div>VB format variabil blocat</div> </div> </div> <div>Valoare implicită este RCF:U</div>
[RCS: <i>mm</i> ]  dimensiunea articolului	<div>parametrul este ignorat pentru RCF:U, F sau V, în celelalte cazuri fiind obligatoriu</div> <div><i>mm</i> reprezintă lungimea utilă a articolului (număr de octeți exprimat în baza 10)</div> <div>Dacă RCF:FB sau D, <i>mm</i> este lungimea fixă a articolului</div> <div>Dacă RCF:VB, <i>mm</i> indică lungimea maximă a articolului</div>



TABELUL VII.2 (continuare)

Parametrul	Descrierea parametrului
[BFS:nn]  dimensiunea zonei tampon	nn este lungimea, în baza 10, a zonei tampon $nn \leq 65535$ Zona tampon trebuie să poată conține înregistrarea fizică maximă din fișier Parametrul e ignorat pentru RCF:B Valoarea implicită este BFS:256
[BFN:p]  numărul de zone tampon	p poate avea valorile 1 sau 2 Valoare implicită BFN:2
[MDO:sss]  organizarea suportului	sss poate fi: STD organizare standard NST organizare nestandard Nu are sens decât pentru banda magnetică Valoare implicită MDO:STD
[SFD]	Parametrul permite în cazul fișierelor exploatate în acces secvențial, organizarea lor secvențială. În absența sa fișierele se consideră de organizare nedefinită. Un fișier organizat secvențial nu poate fi prelucrat decât fie prin citire, fie prin scriere (cifra din indexul fișierului nu poate fi decât 1 sau 2) și admite, dacă este pe suport magnetic, articole de orice format (U, F sau V), dar negrupate în blocuri. În cazul fișierelor pe cartele sau la imprimantă articolele nu pot fi decât de format fix.

Argumentele din listă sînt separate prin virgulă. În scrierea argumentelor nu sînt permise blancurile.

Dacă informațiile cuprinse într-o astfel de instrucțiune depășesc spațiul disponibil pe o cartelă (coloanele 1—72), se pot folosi cartele continue.

Faptul că o descriere de fișiere se continuă pe o altă cartelă se indică printr-un asterisc (\*) în coloana 72. Cartela continuare va avea asterisc în coloana 1, înregistrarea celorlalte informații fiind făcută în coloanele 12 ÷ 72.

Se pot folosi mai multe cartele continue.

Ordinele de descriere a fișierelor se plasează la începutul programului principal, înaintea oricărei alte instrucțiuni din program.

Într-un program pot exista mai multe ordine DEFINE FILE sau USED FILE, care la rîndul lor să descrie unul sau mai multe fișiere.

Singurul caz în care nu se folosesc ordinele DEFINE FILE este acela al fișierelor pe cartele sau la imprimantă, incorporate în fișierele sistemului \*1, \*2.

### 3. FIȘIERE PE CARTELE

Fișierele pe cartele au următoarele caracteristici:

- organizare secvențială;
- organizarea suportului standard;



- articolele sînt de format fix la citire și de format fix sau variabil la scriere, lungimea maximă a unui articol fiind de 80 de caractere;
- ultima cartelă din fișier este o marcă de fișier standard, cu structura:

.EOF

- coloana 1 caracterul.;
- coloanele 2 ÷ 4 caracterele EOF.

Un fișier pe cartele poate fi citit cu instrucțiunea:

READ (*n*, *f*, END = *K*<sub>1</sub>, ERR = *K*<sub>2</sub>) *listă*

unde:

- n* este o constantă întreagă sau o variabilă întreagă a cărei valoare reprezintă numărul FORTRAN al fișierului;
- f* — eticheta unei instrucțiuni FORMAT;
- END și ERR — parametri opționali (cînd ambele opțiuni sînt prezente se scriu în ordinea indicată în READ). END = *K*<sub>1</sub> specifică eticheta *K*<sub>1</sub> a instrucțiunii care preia controlul dacă se citește o marcă de sfîrșit de fișier, iar ERR = *K*<sub>2</sub> specifică eticheta *K*<sub>2</sub> a instrucțiunii care preia controlul în cazul detectării unei erori la transmiterea datelor;
- lista* — o listă de variabile.

Pentru scrierea unui fișier pe cartele se folosește oricare din instrucțiunile de ieșire tratate anterior.

#### 4. FIȘIERE CU ACCES SECVENȚIAL PE SUPORT MAGNETIC

Aceste fișiere pot fi organizate nedefinit sau secvențial. Suportul poate fi organizat standard sau nestandard (în cazul benzilor magnetice).

Articolele din fișier pot fi de format nedefinit, fix sau variabil.

Se admit prelucrările prin scriere, citire sau actualizare prin adăugare de articole la sfîrșitul fișierului.

Prelucrările se realizează cu ajutorul instrucțiunilor:

READ cu FØRMAT  
WRITE cu FØRMAT  
READ fără FØRMAT  
WRITE fără FØRMAT

a. Scrierea și citirea cu FØRMAT. Instrucțiunile de scriere și citire cu FØRMAT sînt:

WRITE (*n*, *f*) *listă*  
și READ (*n*, *f*) END = *K*<sub>1</sub>, ERR = *K*<sub>2</sub>) *listă*,  
descrise anterior.

Fiecărei variabile din lista de intrare/ieșire *i* se rezervă în articol un spațiu egal cu numărul de caractere specificat în descrierea de format. Astfel pentru o variabilă reală descrisă cu F 10.2 se vor rezerva 10 caractere, în care se va înscrie valoarea variabilei.

În cazul *articolelor de format nedefinit* lungimea blocului este lungimea maximă a înregistrărilor FORTRAN pentru fișierele pe bandă, iar pentru fișierele de disc se adaugă acestei lungimi încă 8 octeți corespunzători antetului.



În fișier se scrie numai partea utilă din zona tampon (atunci când lungimea înregistrării este mai mică decât lungimea blocului).

Un exemplu de declarare a unui fișier cu articole de format nedefinit este următorul:

```
* DEFINE FILE A1(DVT : MT, BFS: 40) = 7
```

Fișierul are indexul A, poate fi prelucrat prin citire, este înregistrat pe o bandă magnetică de organizare standard (MDO lipsind se consideră valoarea sa implicită), lungimea blocului este de 40B. Numărul FORTRAN al fișierului este 7. Fișierul este de organizare nedefinită (SFD absent).

Dacă fișierul conține *articole de format fix* dimensiunea zonei tampon este egală cu dimensiunea maximă a înregistrărilor FORTRAN, când suportul este banda magnetică, în cazul fișierelor pe disc adăugându-se 8B pentru antet.

Dacă lungimea unei înregistrări FORTRAN este mai mică decât lungimea stabilită pentru articol și declarată în instrucțiunea de descriere a fișierului, articolul este completat cu spații, în fișier scriindu-se întregul bloc (fig. VII.1).

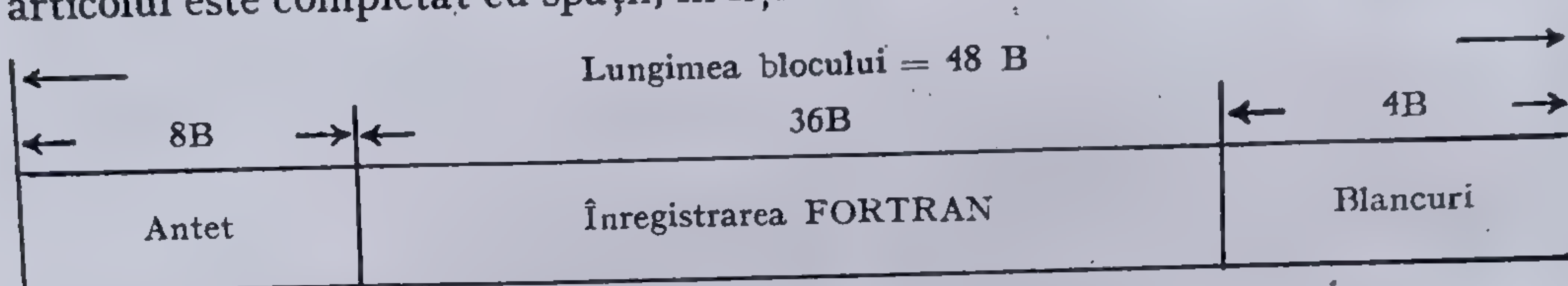


Fig. VII. 1

În cazul utilizării opțiunii SFD articolele sînt automat blocate de către SGF, factorul de blocare fiind stabilit în funcție de dimensiunea zonei tampon (BSF).

Astfel secvența următoare:

```
* DEFINE FILE A2(DVT : RD, BFS : 48, RCF : F) = 2
  WRITE (2,1) A, B, M, N
1  FORMAT (2F10.6, 2I8)
```

definește un fișier pe disc, cu articole de format fix, dimensiunea blocului fiind de 48 B. În acest fișier se scrie un articol, care va cuprinde valorile variabilelor A, B, M, N. Lungimea înregistrării FORTRAN este  $2 \times 10 + 2 \times 8 = 36$  de octeți, deci ultimii 4 octeți ai articolului vor fi completați cu blancuri.

Dacă se consideră exemplul următor:

```
* USED FILE B1 (DVT : RD, RCF : F, BFS : 1024, SFD) = 6
  DIMENSION X (22).
  READ (6, 1) X
1  FORMAT (22 F 10.6)
  WRITE (108,2)X
2  FORMAT ('', 10X, 6 F 12.6)
```

prin USED FILE se declară un fișier pe disc, organizat secvențial. Fișierul a fost creat printr-un program anterior. Articolele din fișier sînt de format fix, dimensiunea unui articol fiind de 220 B. Din fișier se citește primul articol, conținutul său fiind depus în zona X, conform instrucțiunii FORMAT. Vectorul X este apoi afișat la imprimantă.



Se pot organiza fişiere cu acces secvenţial, pe bandă sau disc, în care articolele să fie de *format fix blocat*.

Într-un bloc se grupează  $n$  articole, fiecare bloc avînd acelaşi număr de articole.

Blocul este scris în fişier numai după ce s-au înregistrat  $n$  articole în zona tampon.

De exemplu:

```
* USED FILE A0(DVT : AD, RCS : 60, BFS : 128, RCF : FB) = 10
  DIMENSION B(4), A(6)
  WRITE (10,1) B, C, D, E
1  FORMAT (6E10.4)
  WRITE (10, 1) A
```

Fişierul *A* cu numărul FORTRAN 10 are articole de format fix blocat; factorul de blocare este 2. Fişierul poate fi prelucrat şi prin scriere şi prin citire.

Primul WRITE înregistrează pe fişier un bloc (2 articole); primul articol conţine valorile tabloului *B* şi ale variabilelor *C* şi *D*, al doilea articol conţine valoarea lui *E*, restul fiind completat cu blancuri (fig. VII.2).

Lungimea blocului 128B		
SB	articol 60B	articol 60B
Antet bloc	B(1) B(2) B(3) B(4) C D	E 50 Lancuri

Fig. VII. 2

Cel de-al doilea WRITE introduce în zona tampon un articol conţinînd valorile tabloului *A*. Scrierea pe fişier se va face numai după completarea zonei tampon cu cel de-al doilea articol din bloc (deci după un nou WRITE). Programul FORTRAN poate folosi, în afară de fişierul *A*, numai fişierul sistem \*2, deoarece descrierea de fişiere s-a făcut printr-un ordin USED FILE.

*Articole de format variabil.* Se consideră că fiecare articol formează un bloc. Programatorul trebuie să indice dimensiunea maximă a blocului prin parametrul BFS. Blocul este prevăzut cu un antet de 4B pentru bandă şi 8B pentru disc, în care se înscrie lungimea efectivă a blocului.

Lungimea articolelor fiind diferită, ea trebuie de asemenea menţionată pentru fiecare articol în parte. În acest scop se ataşează articolul cu antet de 4B. Lungimea unui articol este lungimea înregistrării FORTRAN plus antetul (fig. VII.3a).

Lungimea blocului		
lungimea max. art.		
4B	4B	lung. înregistr. FORTRAN
Antet bloc	Antet articol	Înregistrare FORTRAN

Fig. VII. 3 a

În descrierea de fişier, prin parametrul RCS se declară lungimea maximă a unui articol.

Dacă lungimea înregistrării FORTRAN este mai mică decît lungimea maximă a articolului se scrie numai partea utilă a zonei tampon (fig. VII.3b).



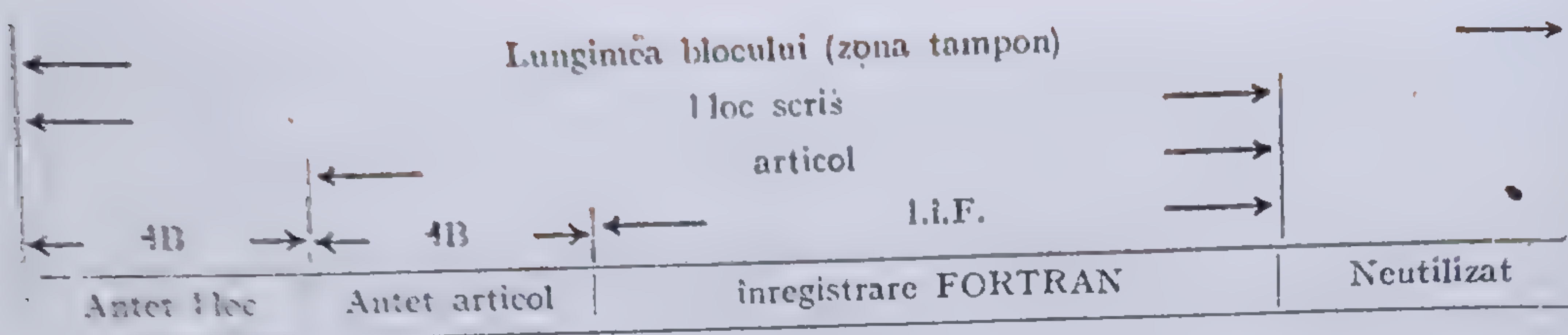


Fig. VII. 3b

Articole de format variabil pot fi blocate. Se pot organiza astfel fișiere care să conțină blocuri de dimensiuni diferite, fiecare bloc avînd un număr variabil de articole.

Dimensiunea zonei tampon va fi dată de factorul de blocare înmulțit cu lungimea maximă a unui articol. Aceasta este lungimea maximă a înregistrării FORTRAN, la care se adaugă  $4B$  antetul articolului, plus antetul blocului ( $4B$  pentru bandă și  $8B$  pentru disc).

Dacă în urma execuției unei instrucțiuni WRITE înregistrarea FORTRAN este mai scurtă decît spațiul înregistrabil din articol (lungimea articolului —  $4$ ) în zona tampon se înscrie numai partea utilă a articolului.

Cînd în zona tampon rămîne un spațiu disponibil mai mic decît RCS se scrie pe fișier blocul; dacă este cazul, partea rămasă din înregistrarea FORTRAN se scrie în blocul următor.

#### Exemplu:

```
* DEFINE FILE A0 (DVT:MT, RCF:V BFS:80) = 15, B0(DVT:RD,
* RCF:VB, RCS:44, BFS:140, BFN:1) = 9 = 3
  DIMENSION V (10), X (6)
  WRITE (15, 1) V
  WRITE (9, 2) A, B, V (2), C
  WRITE (3, 3) (V(I), I = 2, 9), X
1  FORMAT (4F5.2)
2  FORMAT (2F10.4, F5.2, E10.4)
3  FORMAT (8F 5.2)
```

În secvența de program din exemplu se declară fișierele  $A$  și  $B$ : primul pe bandă magnetică, iar al doilea pe disc. Fișierul  $A$  are articole de format variabil, cu lungimea maximă  $72B$  ( $80B - 8B$  antetul blocului și antetul articolului). Fișierul de pe disc conține tot articole de format variabil, blocate.

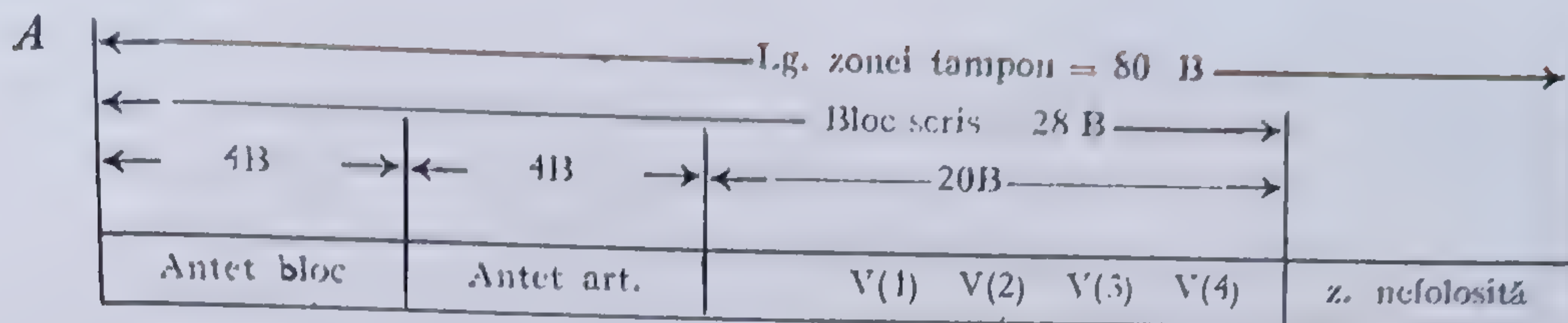
Lungimea maximă a unui articol este de  $44B$ , iar cea a blocului  $140$  octeți. Acest fișier se identifică în programul FORTRAN prin numerele  $9$  și  $3$ . În prima instrucțiune de scriere în fișier referirea la acesta s-a făcut prin numărul  $9$ , în cea de a doua prin numărul  $3$ .

Prezența parametrului BFN:1 în DEFINE FILE indică folosirea unei singure zone tampon. Pentru fișierul  $A$  zona tampon rezervată este de  $80B$ . Prin instrucțiunea WRITE (15,1)  $V$  se scriu în fișier trei blocuri (fig. VII.4a).

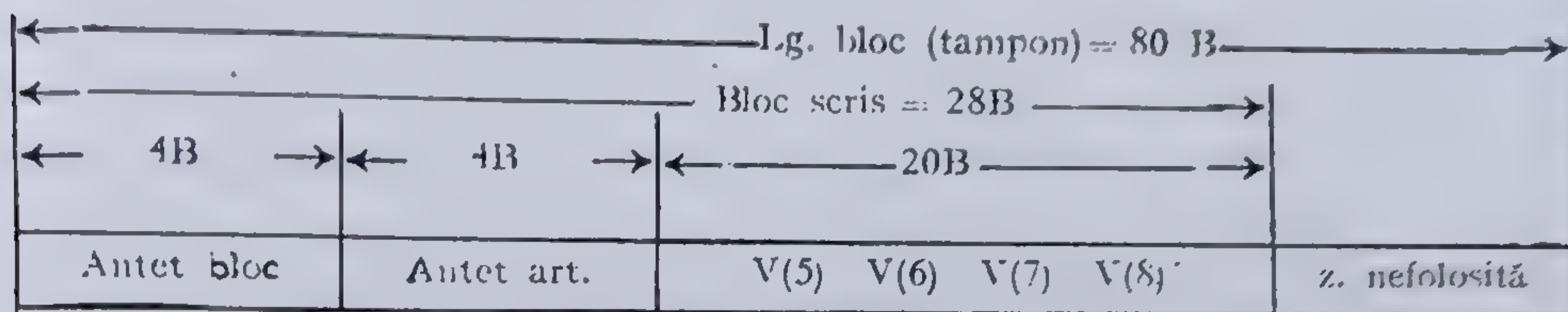
Zona tampon rezervată pentru fișierul  $B$  este de  $140B$ . Cele două instrucțiuni de scriere creează 3 articole în fișier, de dimensiuni  $39B$ ,  $44B$  și respectiv  $34B$ , care sînt depuse în zona tampon. Deoarece aceste articole ocupă  $117B$ , iar spațiul rămas disponibil este mai mic decît  $44B$  (dimensiunea maximă a unui articol) conținutul util al zonei tampon este înregistrat în fișier, unde se creează astfel un bloc de  $117 + 8 = 125 B$  (fig. VII.4 b).



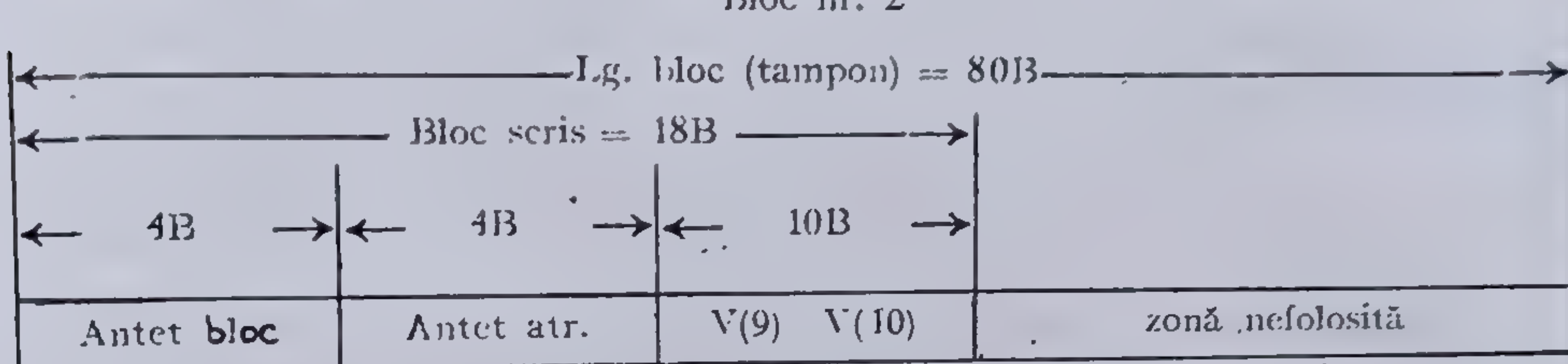
# Fișier



Bloc nr. 1



Bloc nr. 2



Bloc nr. 3

(a)

## Fișier B

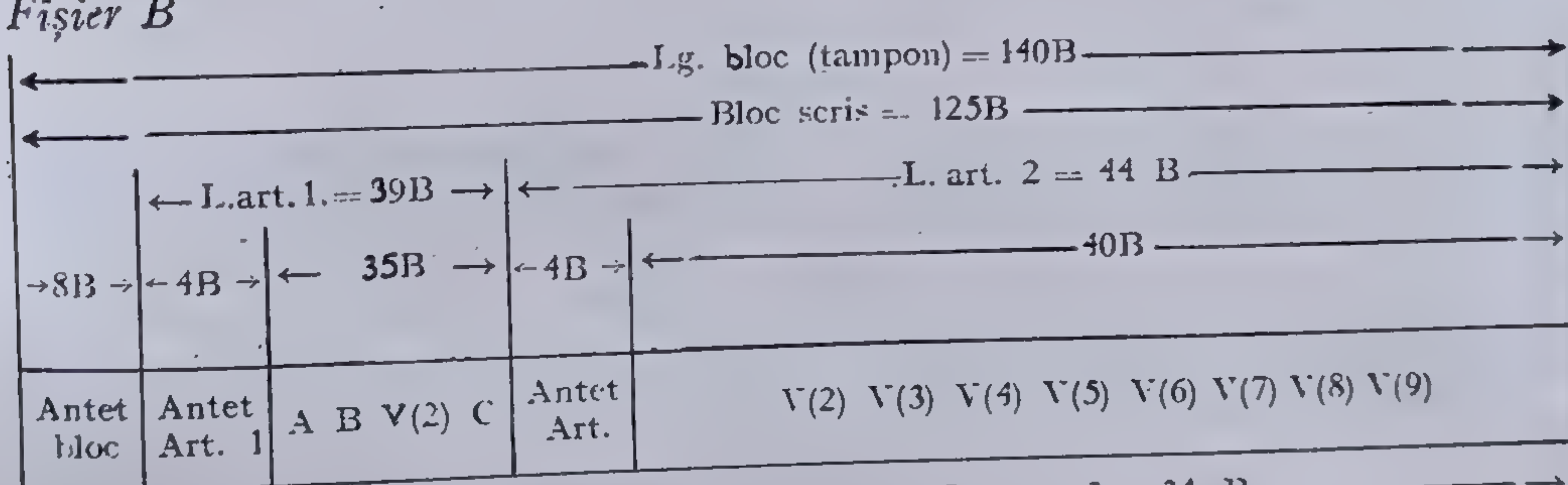
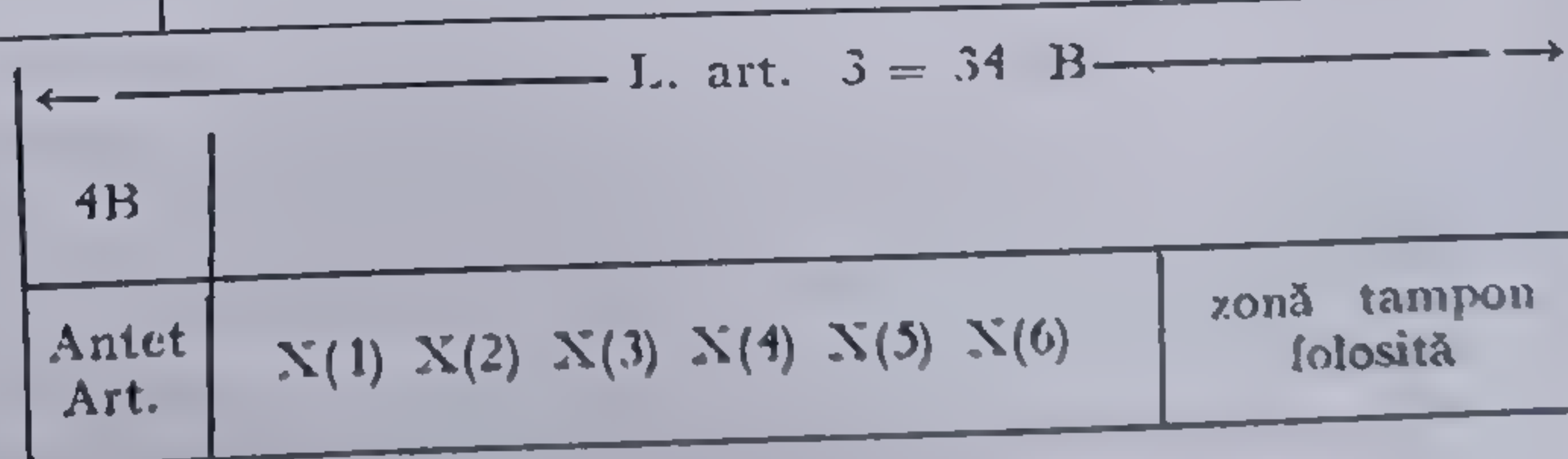


Fig. VII.4.



(b)

b. **Scrierea și citirea fără FØRMAT.** Fișierele cu articole de format variabil, blocate sau nu, admit introducerea și extragera datelor fără descriere de format. Instrucțiunile de intrare/ieșire au în acest caz forma:

READ (n, END = K1, ERR = K2) listă

și

WRITE (n) listă;

unde n, END și ERR au semnificațiile specificate anterior.



La scrierea și citirea cu format, numărul de caractere ocupate de fiecare variabilă era specificat prin descriptorul FORMAT; când nu se folosește descrierea de format, fiecărei variabile  $i$  se rezervă în articol un spațiu conform dimensiunii reprezentării interne (pentru variabile întregi, logice, reale simplă precizie — 4B, pentru variabile reale dublă precizie și variabile complexe — 8B, iar pentru variabile complexe în dublă precizie 16B). Informația care trebuie transmisă este împărțită în seturi de caractere de lungime egală cu lungimea maximă a articolului minus 4 (RCS-4); ultimul set de caractere are lungimea egală sau mai mică decât RCS-4, după cum numărul de caractere din care este constituită informația este sau nu un multiplu de RCS-4.

Fie următoarea secvență:

```
* DEFINE FILE B0(DVT:RD, RCF:VB, RCS:44, BFS:140) = 9
  DIMENSION V(10), X(6)
  WRITE (9), V, X, A, B, C, D, M, N
```

Instrucțiunea WRITE creează pe fișierul cu numărul FORTRAN 9 blocul din figura VII.5.

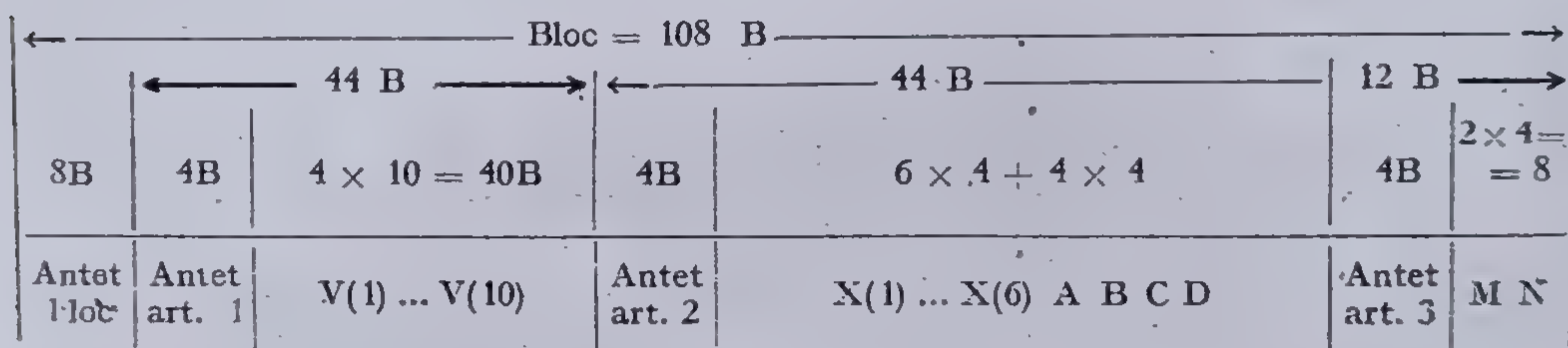


Fig. VII.5

Utilizarea scrierii și citirii fără FORMAT prezintă avantajul reducerii timpului de transfer al unei informații, nemaifiind necesare operațiile de conversie. Astfel, informația odată înregistrată în fișier, la o prelucrare ulterioară nu va trebui convertită în reprezentarea internă.

c. Instrucțiuni auxiliare pentru fișierele cu acces secvențial pe bandă sau disc. Instrucțiunea REWIND are formatul general:

REWIND  $n$

și realizează închiderea fișierului cu numărul FORTRAN  $n$ . Următoarea prelucrare a fișierului (READ sau WRITE), avînd ca prim efect deschiderea fișierului, se va face la nivelul primului articol din fișier.

Dacă ultima prelucrare a fișierului a constatat într-o operație de citire (READ), închiderea fișierului presupune verificarea etichetelor de sfîrșit de fișier și rebobinarea.

Dacă ultima prelucrare a fișierului a constatat într-o operație de scriere (WRITE) sau o instrucțiune ENDFILE pentru o bandă nestandard, închiderea presupune și scrierea de etichete pentru suportul organizat standard sau a două mărci de fișier pentru benzile nestandard.

Instrucțiunea END FILE are formatul general:

ENDFILE  $n$

unde  $n$  este numărul FORTRAN al unui fișier.



Dacă fișierul este pe suport organizat standard, această instrucțiune are ca efect închiderea fișierului. Următoarea instrucțiune READ sau WRITE va realiza deschiderea fișierului și prelucrarea se va efectua începând cu primul articol din fișier.

Dacă suportul este organizat nestandard, instrucțiunea ENDFILE produce scrierea unei mărci de sfârșit de fișier; fișierul rămâne deschis, dispozitivul de citire/scriere fiind poziționat după marca de sfârșit de fișier.

Astfel, dacă se cere să se scrie o marcă de sfârșit de fișier pe o bandă nestandard, să se închidă fișierul și să se poziționeze banda la începutul fișierului, se vor folosi instrucțiunile ENDFILE și REWIND.

**Instrucțiunea BACKSPACE.** La fișierele organizate nedefinit (deci când nu apare parametrul SFD în descrierea de fișier) este posibilă poziționarea dispozitivului de citire/scriere înaintea ultimei înregistrări fizice scrise sau citite.

Acest lucru este important deoarece, în mod normal, nu există acces la o înregistrare din fișier decât parcurgând toate înregistrările anterioare, deci un eventual control al ultimei scrieri nu s-ar putea face imediat.

Pe de altă parte se poate realiza poziționarea fișierului înaintea unui bloc „ENDFILE”, ceea ce permite scrierea în continuare a articolelor în fișier (deci, se extinde fișierul).

Operația de poziționare „înapoi” este declanșată de instrucțiunea

**BACKSPACE *n***

unde *n* este numărul FORTRAN al fișierului prelucrat.

Instrucțiunea este fără efect dacă fișierul este poziționat la început.

**Observație.** Ca și unele instrucțiuni tratate în capitolele precedente, nici instrucțiunea de citire a fișierelor secvențiale, READ cu opțiunea END=K, nu a fost concepută ținând seama de principiile programării structurate. În consecință, pentru a avea certitudinea obținerii unui program structurat în cazul utilizării la intrare a unor fișiere cu acces secvențial, se poate recurge la o variabilă logică, numită, de exemplu EØF, care să permită folosirea structurilor alternative și repetitive, condiția care se testează fiind prezența sau absența unei mărci de sfârșit de fișier. În acest scop EØF are pe toată durata consultării fișierului valoarea .FALSE., iar în momentul întâlnirii unei mărci de sfârșit de fișier i se atribuie valoarea .TRUE., prin instrucțiunea care urmează instrucțiunii CONTINUE cu eticheta K. Prin urmare, instrucțiuni de tipul IF(EØF)GOTO ... sau IF(.NOT.EØF)GOTO ... permit simularea tuturor structurilor repetitive cunoscute.

Pentru problemele care prelucrează fișiere în acces secvențial este tipică utilizarea structurii repetitive condiționate intern (condiția din IF este EOF).

## 5. FIȘIERE CU ACCES DIRECT

În cazul fișierelor depuse pe orice tip de disc este posibil accesul direct la informații. Fișierele cu acces direct sînt organizate nedefinit. Articolele sînt de lungime fixă, grupate în blocuri fără antet (fig. VII.6).

Fiecărui articol din fișier i se atașează un număr: prin program se indică numărul articolului ce trebuie prelucrat. Acest lucru dă posibilitatea transmiterii datelor în ordinea dorită.



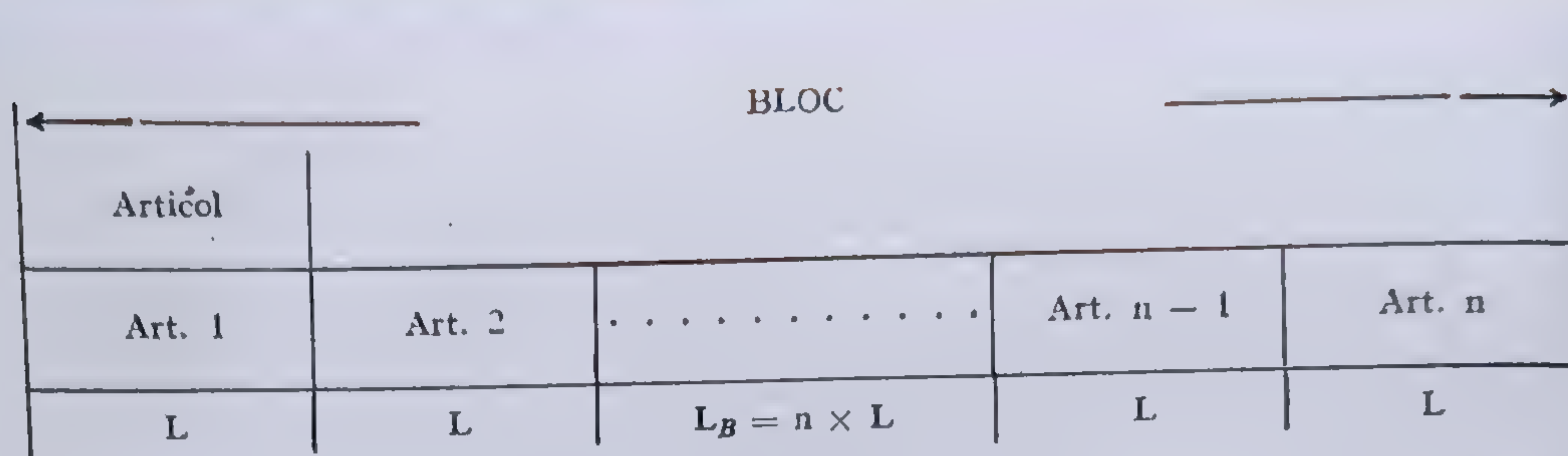


Fig. VII.6

Operațiile de intrare/ieșire se realizează cu ajutorul instrucțiunilor READ și WRITE cu acces direct, în care, spre deosebire de cele pentru acces secvențial, se specifică pe lângă numărul FORTRAN al fișierului și numărul articolului care se prelucrează.

Scrierea sau citirea articolelor din fișier se poate face cu sau fără format.

Execuția unei instrucțiuni READ sau WRITE provoacă transferul unor informații între zona tampon și fișier, începînd cu informațiile conținute în articolul al cărui număr se specifică în instrucțiune. Numărul de articole prelucrate printr-o astfel de instrucțiune depinde de numărul elementelor din lista de intrare/ieșire și, dacă este cazul, de specificațiile de format. Sistemul furnizează, la cerere, numărul articolului care urmează ultimului articol prelucrat.

Lungimea blocului se alege astfel încît să poată cuprinde cea mai mare înregistrare FORTRAN.

Instrucțiunile READ și WRITE pentru fișierele cu acces direct sînt de forma:

READ ( $n'i, v, f, \text{ERR} = d$ ) listă

și

WRITE ( $n'i, v, f$ ) listă

unde:

- |                  |   |  |
|------------------|---|--|
| $n$              | — | reprezintă numărul FORTRAN al fișierului;  |
| $i$              | — | o expresie întreagă pozitivă a cărei valoare indică poziția relativă a articolului ce trebuie citit sau scris (numărul articolului în fișier), $i$ este despărțit de primul argument printr-un apostrof (dacă ar fi fost despărțit printr-o virgulă ar fi putut fi confundat cu o etichetă de FORMAT); |
| $f$              | — | argumentul opțional care reprezintă eticheta unei instrucțiuni FORMAT;   |
| $v$              | — | este opțional și reprezintă o variabilă întreagă, numită variabilă asociată, a cărei valoare, la sfîrșitul execuției operației de intrare/ieșire, coincide cu numărul articolului care urmează ultimului articol prelucrat;  |
| $\text{ERR} = d$ | — | o specificație opțională, care indică eticheta instrucțiunii executabile la care se transferă controlul în cazul unei erori de transmitere a datelor.  |

Exemplul 1:

```
*DEFINE FILE A1 (DVT:RD, RCF:D, RCS:50, BFS:150) = 9
  DIMENSION V(30)
  READ (9'50,K, 1) V
1  FORMAT (10F5.0)
```



În această secvență de program instrucțiunea READ citește datele pentru  $V$  din fișierul cu numărul FORTRAN 9, începînd cu articolul cu numărul 50. Deoarece, conform instrucțiunii FORMAT o înregistrare FORTRAN conține 10 elemente din  $V$  și ocupă  $10 \times 5 = 50$  octeți, pentru a obține datele pentru întreg vectorul  $V$  se vor citi 3 articole din fișier (1 articol furnizează datele pentru 10 elemente din  $V$ ). Variabilele  $V(1)$ ,  $V(2)$  ...  $V(10)$  primesc valori din articolul 50,  $V(11)$ ,  $V(12)$  ...  $V(20)$  din articolul 51, iar  $V(21)$ ,  $V(22)$  ...  $V(30)$  din articolul 52. La sfîrșitul operației de citire  $K$  va avea valoarea  $50 + 3 = 53$ .

*Exemplul 2:*

```
* DEFINE FILE A2 (DVT:MD, RCF:D, RCS:30, BFS:300) = 6
  DIMENSION M(10)
  DO 2 I = 1,10
    READ (105,1)M
1    FORMAT (10I3)
    WRITE (6'I,1)M
2    CONTINUE
```

Conform formatului o înregistrare FORTRAN are  $3 \times 10 = 30B$ , deci va ocupa un articol în fișier.

Scrierea se face începînd cu articolul 1, în care se înregistrează valorile  $M(1)$ ,  $M(2)$  ...  $M(10)$ ; se scriu 10 articole.

*Exemplul 3:*

```
* DEFINE FILE B0(DVT:MD, RCF:D, RCS:40, BFS:80) = 7
  DIMENSION A (20)
  DO 1 I = 1,100
    READ (105,2) A
2    FORMAT (10 F 8.4)
    WRITE (7'I) A
1    CONTINUE
```

Scrierea pe disc se face fără FORMAT. Fiecare element din  $A$  va ocupa 4B, deci pentru scrierea întregului vector se vor înregistra  $4 \times 20 = 80 B$ , adică două articole.

Astfel, dacă se înregistrează elementele unei matrici cu 20 coloane și 100 de linii, se introduce pe rînd în memoria centrală cîte o linie în vectorul  $A$ , matricea urmînd să fie în întregime înregistrată pe disc.

În cazul fișierelor cu acces direct, o operație de intrare/ieșire poate fi pregătită poziționînd dispozitivul de citire/scriere pe acel cilindru de pe disc corespunzător articolului care urmează a fi prelucrat. Aceasta se realizează prin instrucțiunea FIND:

FIND ( $n'i$ )

unde  $n$  și  $i$  au aceeași semnificație ca și în cazul instrucțiunilor READ și WRITE cu acces direct.

Instrucțiunea este efectivă numai dacă instrucțiunea de citire/scriere care urmează se referă la articolul cu numărul  $i$ , în acest caz operația de intrare/ieșire se execută mai rapid.



### Exemplul 1:

```
DEFINE FILE B1(DVT:RD, RCF:D, RCS:20, BFS:240) = 8  
  DIMENSION M(24)  
  FIND (8'25)  
  .  
  .  
  .  
  READ (8'25)M
```

Simultan cu execuția instrucțiunilor ce urmează după FIND (8'25) și preced pe READ (8'25)M se caută și se găsește articolul cu numărul 25 din fișierul cu numărul FORTRAN 8.

### Exemplul 2:

```
  DIMENSION V(7), W(7)  
  READ (105,2) I, V  
2  FORMAT (I2, 7F 10.2)  
  WRITE (8'I,3)V  
3  FORMAT (7F10.2)
```

```
  .  
  .  
  .  
  .  
  I = 4.5*A - B/C  
  FIND (8'I)
```

```
  .  
  .  
  W(K) = (A + B)*K
```

```
  .  
  .  
  WRITE (8'I,3)W
```

Se observă că folosirea instrucțiunii FIND nu are rost decât în cazul celei de a doua instrucțiuni de scriere în fișier.

## APLICAȚII ȘI PROBLEME

**Programul VII.1.** Se consideră un fișier pe cartele FCART relativ la produsele care intră într-o magazie. Pe fiecare cartelă s-au înregistrat

- codul produsului → ICOD
- denumirea produsului → DEN
- cantitatea introdusă în magazie → CANT
- prețul unitar pentru produsul respectiv → PU.

Macheta cartelei este descrisă în figura VII.7a.

În FCART pot exista mai multe cartele cu același cod produs (în momente diferite s-au introdus în magazie anumite cantități dintr-un produs). În urma unei sortări prealabile, cartelele cu același cod sînt în secvență.





a.

```

* * * * *
*          *                               *
*   COD    *   DENUMIRE  PRODUS          *   COST TOTAL   *
*          *                               *
* * * * *
*          *                               *
* - - - - - * - - - - -                  * - - - - - *

```

b.

Fig. VII.7

Se propune ca pentru fiecare produs să se determine valoarea întregii cantități aflate în magazie și să se listeze o situație cu aceste rezultate (fig. VII.7b).

Pentru rezolvarea problemei se va prelucra pe rând fiecare cartelă din FCART. Valoarea întregii cantități dintr-un anumit produs se depune în TVAL. Deoarece pot fi mai multe cartele cu același cod, pentru acestea în TVAL vor trebui cumulate valorile  $CANT \times PU$ . Pentru a compara codul actual cu cel citit anterior se folosește o zonă de manevră care păstrează, la un moment dat, prima înregistrare cu codul diferit de codul anterior.

În figura VII.8 sunt descrise schema de prelucrare și schema logică conform căreia se va realiza programul VII.1.

Programul VII.2. Se dă un fișier pe cartele reprezentând operații CEC, cu macheta descrisă în figura VII.9a.

Se cere crearea unui fișier pe bandă care să reproducă fișierul pe cartele.

S-a considerat că fișierul de pe cartele este nevid. Programul se poate completa ușor pentru a ține seama și de situația în care fișierul este vid (programul VII.1).

În figura VII.9b și c sunt descrise schema de prelucrare și schema logică a programului de creare a fișierului (programul VII.2).

Programul VII.3. Să se consulte fișierul pe bandă creat în programul anterior și să se obțină o listă cu codul și numele persoanelor care au soldul mai mic de 100 de lei.

Schema logică este descrisă în figura VII.10. Pe baza acestei scheme s-a elaborat programul VII.3.

Programul VII.4. Să se actualizeze fișierul 'SOLDURI' creat în programul VII.2, prin adăugarea de noi articole de același tip.

Pentru a putea adăuga articolele, fișierul 'SOLDURI' trebuie mai întâi citit în întregime, iar apoi, folosind instrucțiunea BACKSPACE, poziționat înaintea mărcii de sfârșit de fișier.

Programul este VII.4.

Programul VII.5. Cu datele conținute în fișierul 'SOLDURI' de pe MT1 să se creeze un fișier pe disc, în acces direct, fiecare articol conținând cimpurile ICOD, NUME și cimpul SOLD actualizat ( $SOLD = SOLD + DEP - REST$ ).

Pentru fiecare articol numărul său este dat de ICOD, deci o instrucțiune de scriere pe disc va fi de forma:

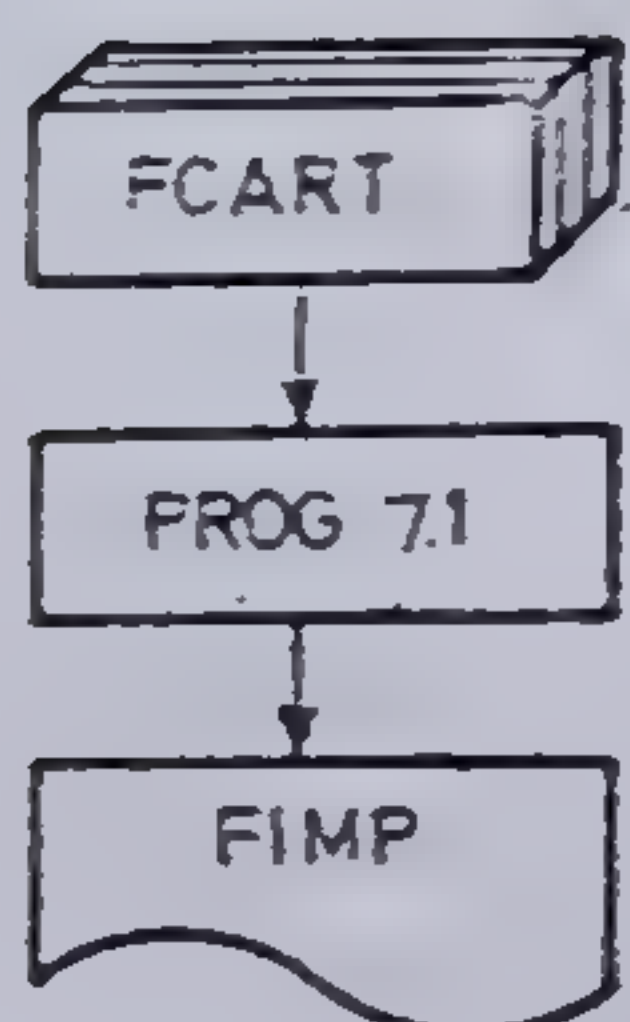
WRITE (n'ICOD, f) ICOD, NUME, SOLD

Astfel, accesul direct la articol se face pe baza valorii cimpului ICOD.

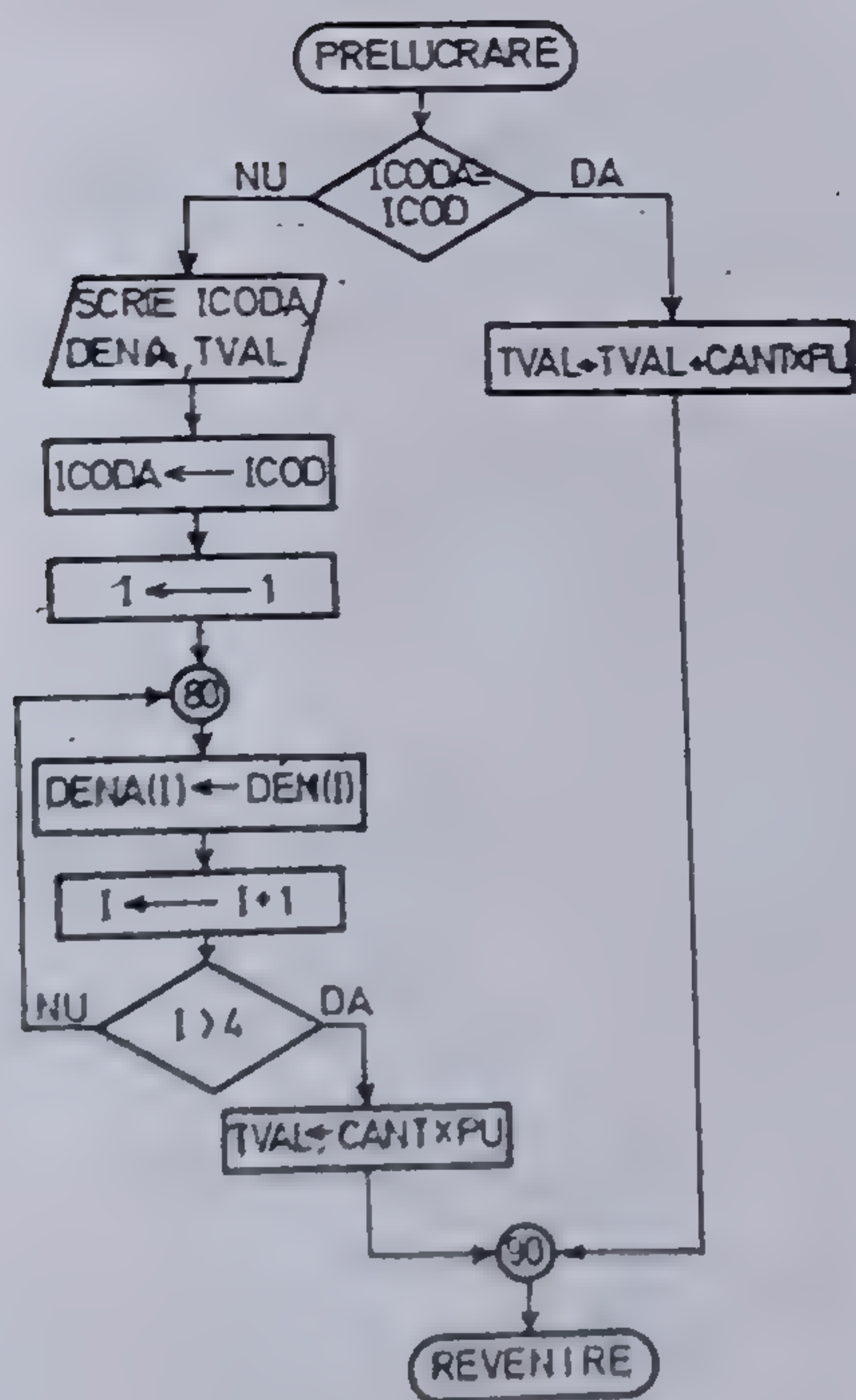
O astfel de scriere va permite ulterior completarea fișierului prin inserarea de noi articole.

Programul este VII.5.

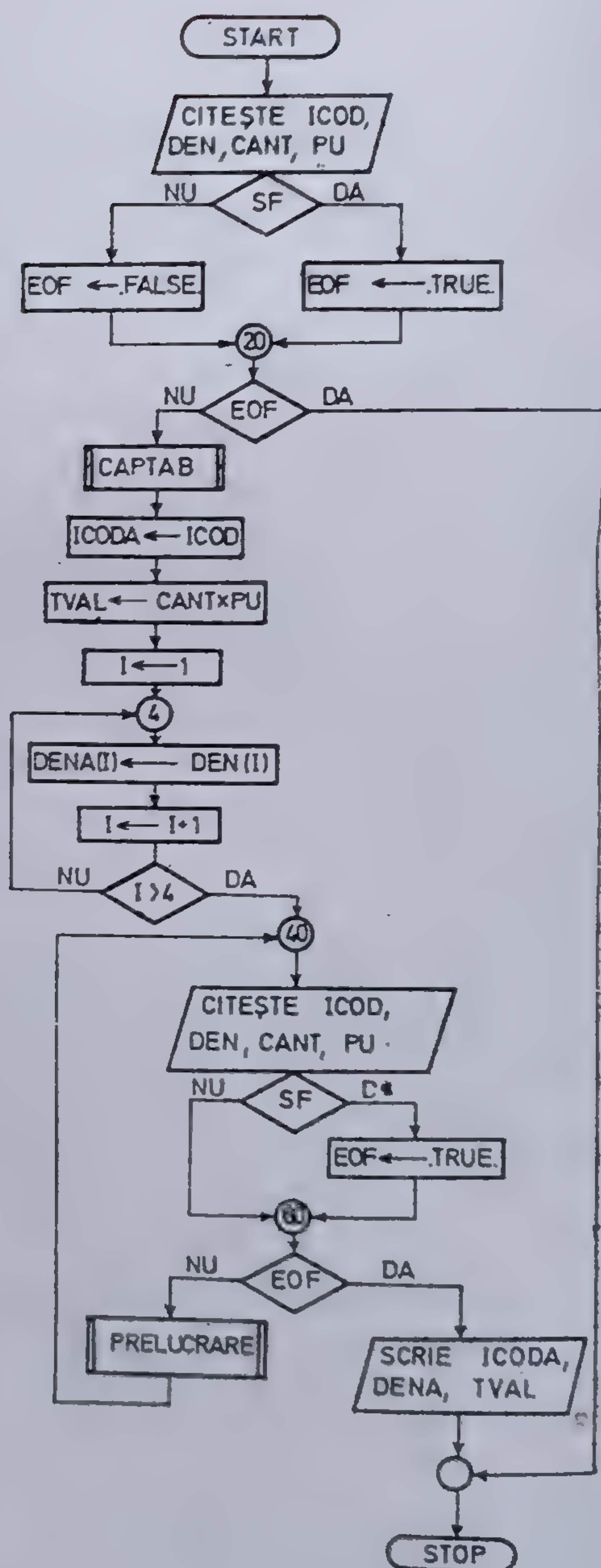




a.



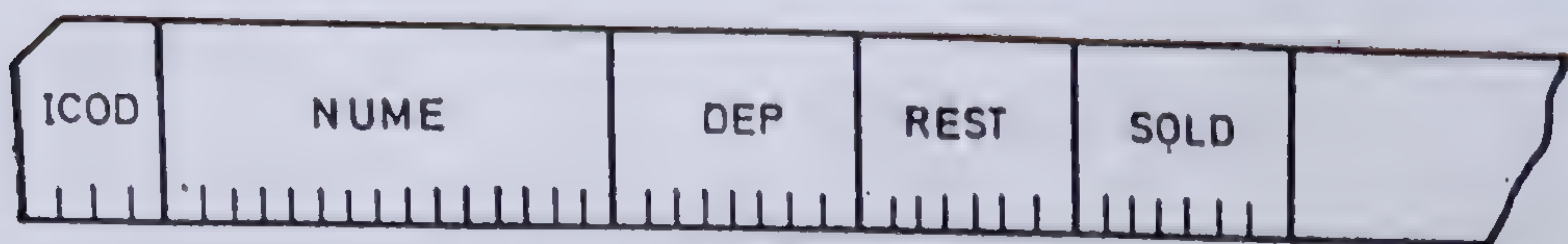
c.



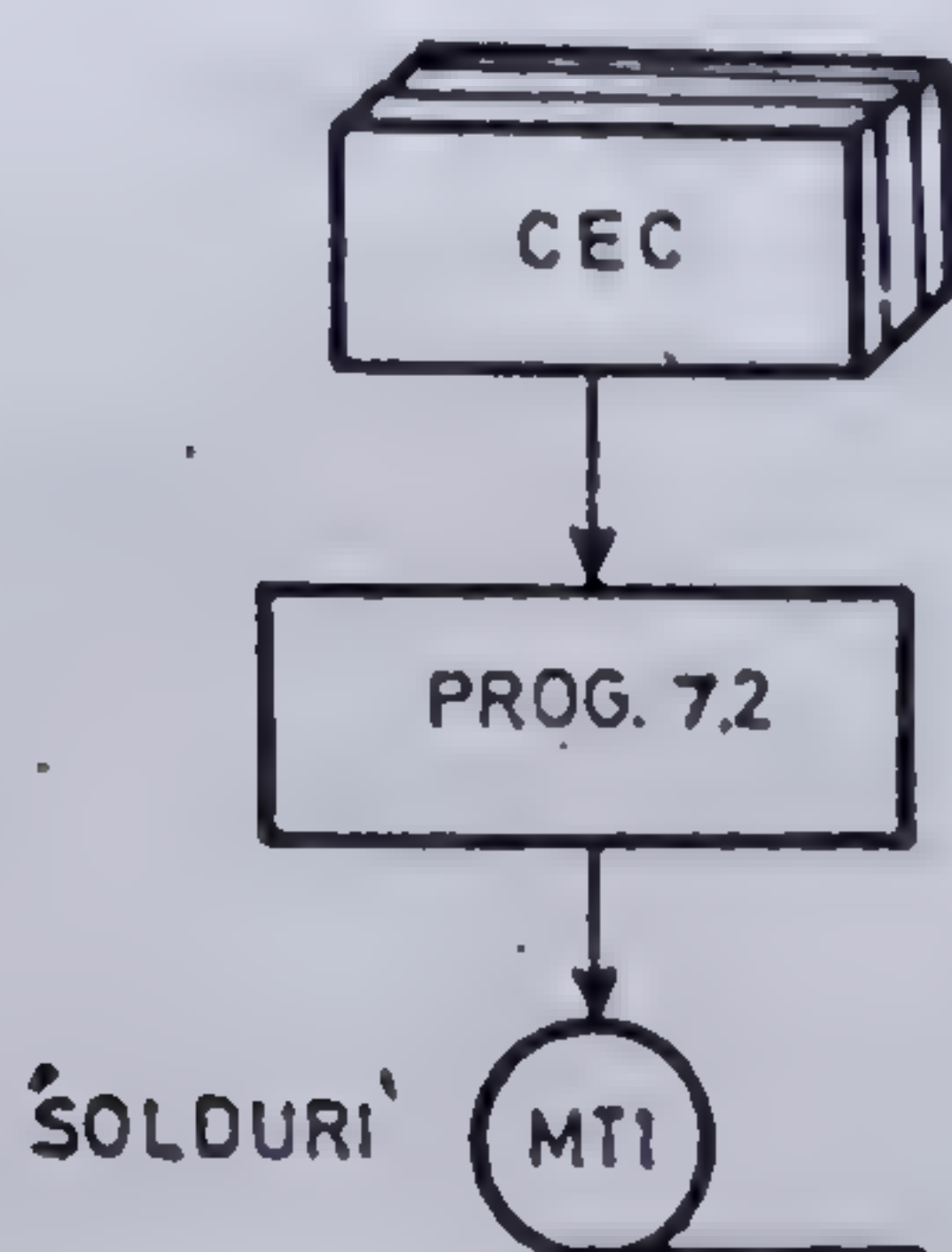
b.

Fig. VII.8

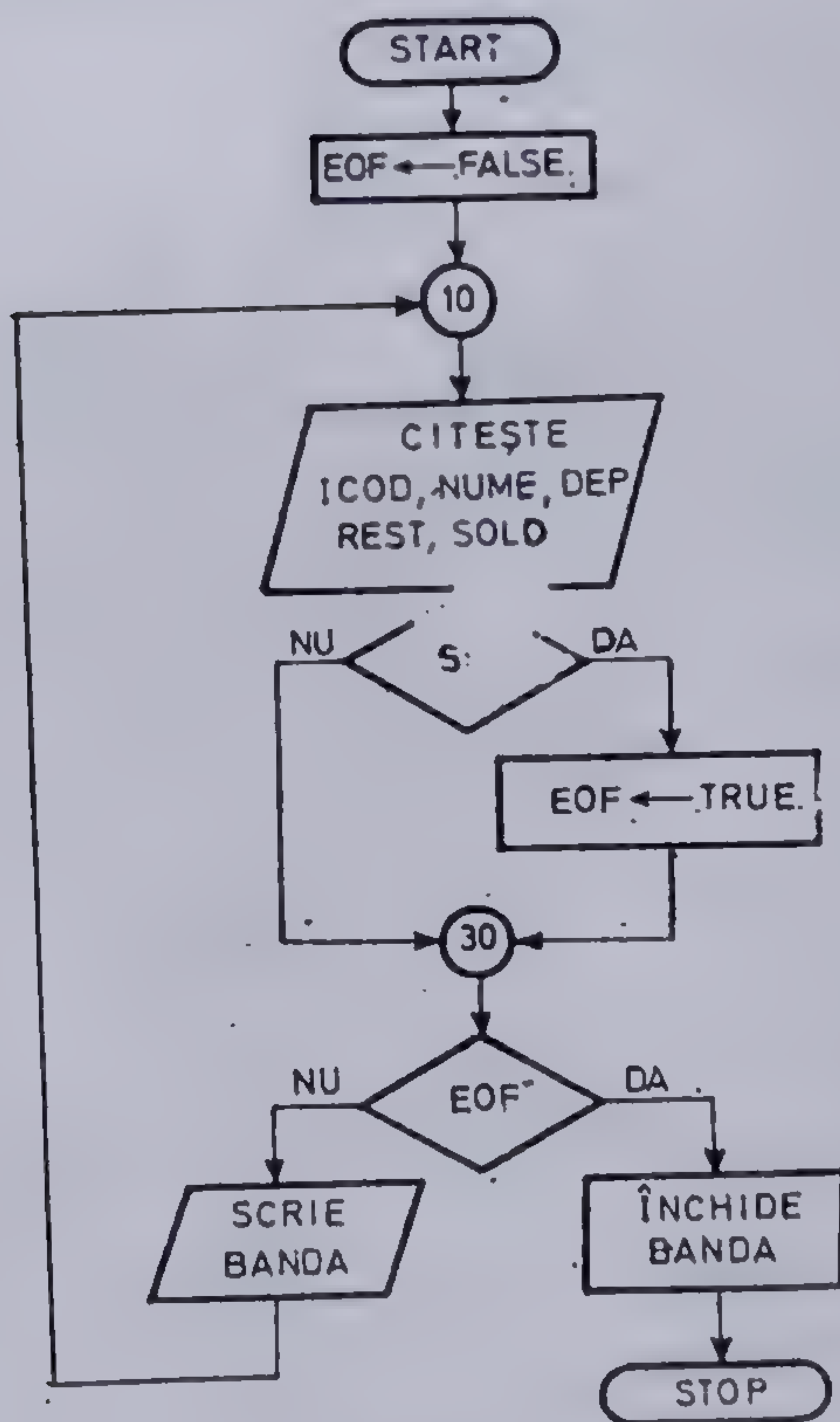




a.



b.



c.

Fig. VII.9



```

•      UOB FISIER2,AN:II00,PN:IOANA
•      COMPILE FORTRAN
C
C      ****
C      *  PROGRAM DE CREARE A UNUI FISIER      *
C      *          PE BANDA MAGNETICA          *
C      ****
C
*  DEFINE FILE A2(DVT:MT,RCS:50,BFS:1000,RCF:FB)=5
    DIMENSION NUME(4)
    LOGICAL EOF
    EOF=.FALSE.
10  CONTINUE
    READ(105,1,END=20) ICOD,NUME,DEP,REST,SOLD
    1  FORMAT(I4,4A4,3F9.2)
    GO TO 30
20  CONTINUE
    EOF=.TRUE.
30  CONTINUE
    IF(EOF) GO TO 100
    WRITE(5,2) ICOD,NUME,DEP,REST,SOLD
    2  FORMAT(I4,4A4,3F10.2)
    GO TO 10
100 CONTINUE
    ENDFILE 5
    STOP
    END

•      LINK
•      INIT DVT:MT,VS:MT1MAN
•      ASSIGN A,DVT:MT,VS:MT1MAN
•      LABEL A, FN: 'SOLDURI'
•      RUN TIME:20,NL:1000
15  SANDU ADELA          00      100000      52500
20  MANASE ILEANA       00      13750      20000
1000 DUMITRU MARIN     500000      00      10000
907  AVRAM MARIA       70000      000      10000
• EOF
•      EOJ

```

PROGRAMUL VII.2



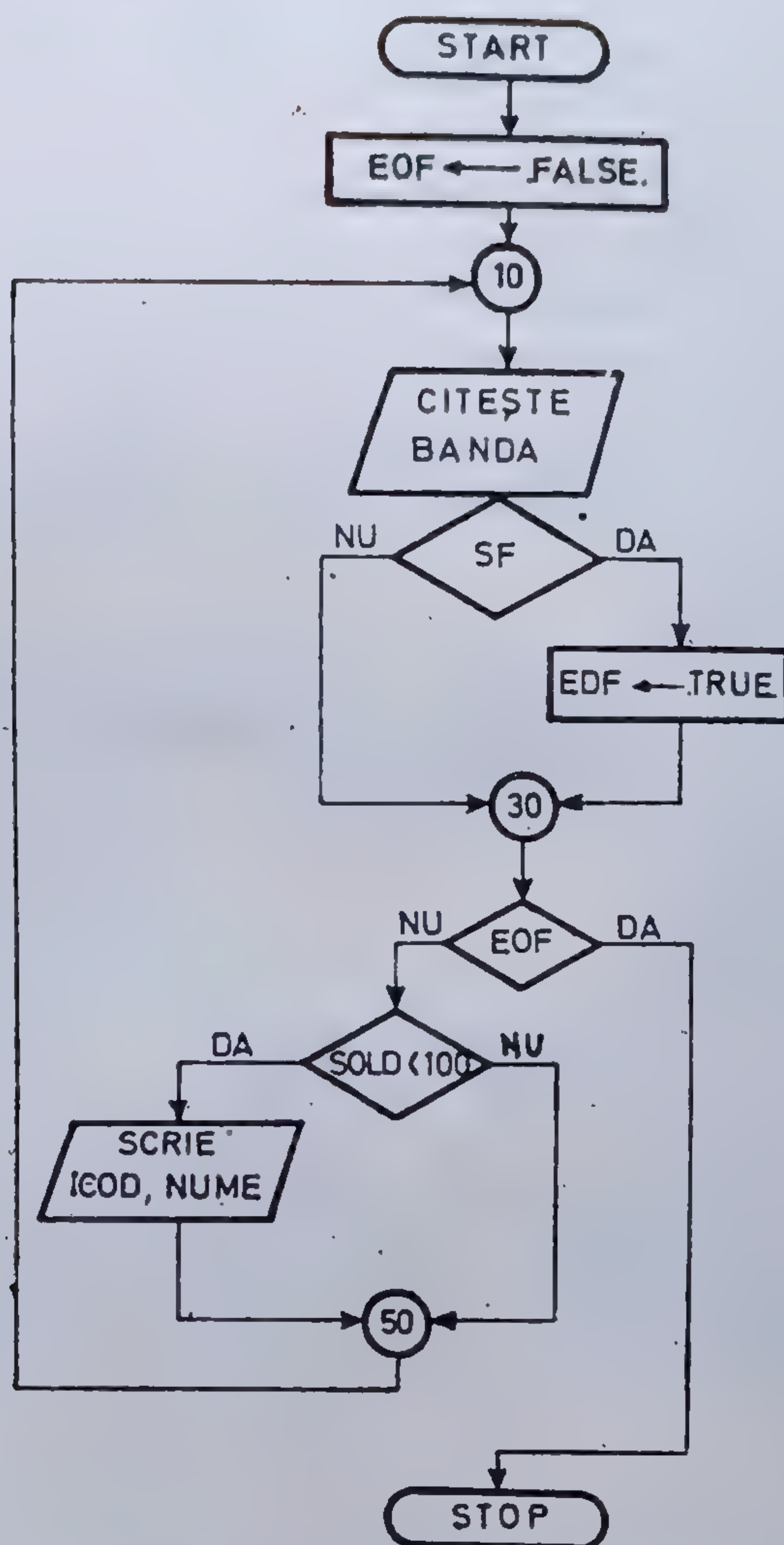


Fig. VII.10

## PROBLEME.

1. Se consideră o matrice reală  $A$ . Să se creeze un fișier pe bandă care să conțină, pentru fiecare element al matricei, următoarele informații:
  - valoarea elementului;
  - valoarea absolută a elementului;
  - rădăcina pătrată din valoarea absolută;
  - pătratul elementului;
  - puterea a treia a elementului;
2. Să se prelucreze fișierul creat pe bandă prin programul anterior, extrăgând la imprimantă toate înregistrările care conțin informații referitoare la numere cuprinse în intervalul  $[-50, +151.25]$ .
3. Să se creeze un fișier pe bandă cu înregistrări care au 7 câmpuri alfanumerice, fiecare cu câte 4 baiți.

*Indicație.* Pentru acest fișier se alege indexul A2 (s-a luat 2 deoarece este o creare de fișier). Drept-număr pentru fișier se la 1. Pentru a putea crea fișierul este nevoie de un vector  $X$  cu componente pentru a scrie din memoria centrală pe bandă.



JOB FISIER3,AN:1100,PN:IOANA  
 COMPILE FORTRAN

\*\*\*\*\*  
 \* PROGRAM DE CONSULTARE A FISIERULUI \*  
 \* PE BANDA CREAT IN PROGRAMUL 7.2 \*  
 \*\*\*\*\*

\* DEFINE FILE A1(DVT:MT,RCF:FB,RCS:50,BFS:1000)=5  
 DIMENSION NUME(4)  
 LOGICAL EOF  
 EOF=.FALSE.

10 CONTINUE  
 READ(5,1,END=20) ICOD,NUME,DEP,REST,SOLD  
 1 FORMAT(I4,4A4,3F10.2)  
 GO TO 30  
 20 CONTINUE  
 EOF=.TRUE.  
 30 CONTINUE  
 IF (EOF) GO TO 40  
 IF (.NOT.(SOLD.LT.100.)) GO TO 50  
 THEN  
 WRITE(108,2) ICOD,NUME  
 2 FORMAT(' ',10X,I4,4X,4A4)  
 50 CONTINUE  
 GO TO 10  
 40 CONTINUE  
 STOP  
 END

LINK  
 ASSIGN A,DVT:MT,VS:MT1MAN  
 LABEL A,FN:'SOLDURI'  
 RUN TIME:20  
 EOJ

PROGRAMUL VII.3

4. Să se facă un program pentru scrierea la imprimantă a fișierului creat anterior.
5. Se consideră un fișier pe cartele. Pe fiecare cartelă sînt scrise trei numere  $N, A, B$  ( $1 \leq N \leq 5$ ) cu care se definesc funcțiile

$$F(N) = \begin{cases} A + B \\ A - B \\ A * B \\ \max(A, B) \\ \min(A, B) \end{cases}$$

dacă  $N = 1$   
 dacă  $N = 2$   
 dacă  $N = 3$   
 dacă  $N = 4$   
 dacă  $N = 5$

$$G(N) = \begin{cases} B^2 \\ A^2 \end{cases}$$

dacă  $N$  este impar  
 dacă  $N$  este par



```

.      JOB FISIER4,AN:1100,PN:IOANA
.      COMPILE FORTRAN
C      *****
C      *      PROGRAM DE ACTUALIZARE A FISIERULUI *
C      *      'SOLDURI' PRIN ADAUGAREA UNOR AR- *
C      *      TICOLE LA SFIRSITUL BENZII.      *
C      *****
C
* DEFINE FILE A0(DVT:MT,RCF:FB,RCS:50,BFS:1000)=5
  DIMENSION NUME(4)
  LOGICAL EOF
  EOF=.FALSE.
10  CONTINUE
    READ(5,1,END=20) ICOD,NUME,DEP,REST,SOLD
    1  FORMAT(I4,4A4,3F10.2)
    GO TO 30
20  CONTINUE
    EOF=.TRUE.
30  CONTINUE
    IF(EOF) GO TO 40
    GO TO 10
40  CONTINUE
    BACKSPACE 5
    EOF=.FALSE.
50  CONTINUE
    READ(105,2,END=60) ICOD,NUME,DEP,REST,SOLD
    2  FORMAT(I4,4A4,3F9.2)
    GO TO 70
60  CONTINUE
    EOF=.TRUE.
70  CONTINUE
    IF(EOF) GO TO 80
    WRITE(5,1) ICOD,NUME,DEP,REST,SOLD
    GO TO 50
80  CONTINUE
    ENDFILE 5
    STOP
    END

.      LINK
.      ASSIGN A,DVT:MT,VS:MT1MAN
.      LABEL A,FN:'SOLDURI'
.      RUN TIME:20
.
90  BRAN ILIE          40000          00      140000
1500 STAN GORE        1560000         00      2244000
2803 IVAN ANA          00          50000      850000
.EOF
.      EOJ          PROGRAMUL VII.4

```



JOB FISIER5,AN:1100,PN:IOANA  
COMPILE FORTRAN

```

*****
* PROGRAM DE CREARE A UNUI FISIER *
* IN ACCES DIRECT *
*****
* DEFINE FILE B2(DVT:AD,RCF:D,RCS:30,BFS:1024)=7
* DEFINE FILE A1(DVT:MT,RCF:FB,RCS:50,HFS:1000)=5
  LOGICAL EOF.
  DIMENSION NUME(4)
  EOF=.FALSE.
10 CONTINUE
  READ(5,1,END=20) ICOD,NUME,DEP,REST,SOLD
  1  FORMAT(I4,4A4,3F10.2)
  GO TO 30
20 CONTINUE
  EOF=.TRUE.
30 CONTINUE
  IF(EOF) GO TO 40
  SOLD=SOLD+DEP-REST
  WRITE(7,1) ICOD,NUME,SOLD
  2  FORMAT(I4,4A4,F10.2)
  GO TO 10
40 CONTINUE
  STOP
  END

  LINK
  ALLOC DVT:AD,VS:AD1MAN,FM:'DISC',
        AM:ANY,SZ:5
  ASSIGN B,DVT:AD,VS:AD1MAN
  LABEL B,FM:'DISC'
  ASSIGN A,DVT:MT,VS:MT1MAN
  LABEL A,FM:'SOLDURI'
  RUN TIME:20
  EOJ

```

#### PROGRAMUL VII.5

Cu ajutorul acestor funcții să se calculeze

$$C = \begin{cases} G(1), G(2) \dots G(N) & \text{dacă } N = 3 \\ G(1) \dots G(N), P(1) \dots P(N) & \text{dacă } N \neq 3 \end{cases}$$

Să se afișeze valorile C obținute.

6. Să se găsească numerele prime mai mici ca 100. Acestea vor fi organizate ca fișier secvențial pe bandă magnetică.

După ce va fi creat, fișierul se va lista la imprimantă.



*Indicație.* La determinarea numerelor prime se va folosi proprietatea că un număr  $M$  este prim dacă nu este divizibil cu numerele mai mici ca  $M/2$  (vor fi analizate numai numerele impare în acest sens).

7. Fie  $N(4,8)$  o matrice de numere întregi. Să se creeze un fișier pe disc care să fie utilizat (exploatat) în acces direct. Un articol din fișier va conține un element al matricei  $N$ . Pentru a controla dacă s-au scris în fișier toate înregistrările se va utiliza variabila  $K$  a cărei valoare va da înregistrarea la care se oprește scrierea (în acest caz  $K = 33$ , deoarece fișierul va conține  $4 \times 8 = 32$  înregistrări).

Primele două linii ale matricii  $N$  sînt pe o cartelă de date, iar celelalte două pe o altă cartelă de date.

8. Fișierul creat în exercițiul anterior va fi utilizat după cum se precizează în continuare. Anumite elemente ale matricei, deci unele înregistrări ale fișierului, se vor modifica. Aceste elemente se vor preciza pe cîte o cartelă prin cei doi indici  $(I, J)$  și prin valoarea ce o capătă elementul respectiv.

Pentru identificarea înregistrării ce urmează să fie modificată se va utiliza o variabilă  $K$  a cărei valoare rezultă din  $I$  și  $J$  astfel:

$$K = (I - 1) \cdot 8 + J$$

După modificarea a 4 elemente se va lista, la imprimantă, tot fișierul.

9. Se consideră trei vectori  $A, B, C$ , fiecare cu cîte 5 componente și două variabile notate cu  $E, F$ . Cu aceste date ale căror valori se vor preciza printr-o instrucțiune DATA, să se creeze un fișier secvențial pe bandă magnetică, folosind instrucțiunea WRITE fără FORMAT (informația din memorie trece pe bandă sub forma în care este reprezentată în memorie). Exploatarea fișierului se va face cu ajutorul lui READ, fără FORMAT, aducînd cei trei vectori pe rînd în vectorul numit  $X$ , care are 15 componente. Cele două valori  $E$  și  $F$  vor fi aduse în memorie, la adresele  $Y(1)$  și respectiv  $Y(2)$ .

Se va considera că:

- vectorii  $A$  și  $B$  sînt numerici (componentele lor au ca valori numere reale);
- vectorul  $C$  este alfanumeric, componentele lui avînd valorile specificate prin DATA;
- variabilele  $E$  și  $F$  sînt logice.

10. Să se scrie un program în care să se genereze numerele reale în progresie aritmetică cu rația  $R$ , cuprinse în intervalul  $[A, B]$ . Valorile  $R, A, B$  se citesc de pe o cartelă sau se vor inițializa prin DATA. Pentru fiecare număr se va crea o înregistrare într-un fișier pe bandă. Înregistrarea va cuprinde:

- numărul;
- rădăcina pătrată din valoarea absolută a numărului;
- pătratul numărului;
- partea întreagă a numărului.

Fișierul astfel creat va fi prelucrat în vederea obținerii la imprimantă a unui tabel care să conțină înregistrările pentru care numărul inițial este cuprins într-un interval  $[C, D] \subseteq [A, B]$ . Valorile  $C, D$  se citesc de pe o cartelă.

Se verifică  $C \geq A$  și  $D \leq B$ . În caz contrar se va afișa la imprimantă un mesaj.

11. Fiînd dat fișierul în acces direct creat în programul VII.5 să se actualizeze fișierul prin modificare și/sau adăugare de articole.

Tipul actualizării este dat de o cartelă parametru care conține una din valorile posibile ale variabilei PARAM:

PARAM = 1 actualizare prin modificarea cîmpului SOLD din anumite articole

PARAM = 0 modificare și adăugare

PARAM = 2 adăugare de noi articole.

Modificarea cîmpului SOLD se face pentru articolele care au același cod cu articolele dintr-un fișier pe cartele. Cartelele conțin ICOD, NUME, DEP și REST. Pe baza valorilor înregistrate în cîmpurile DEP sau REST soldul vechi este actualizat:  $SOLD = SOLD + DEP - REST$  (evident, unul din cîmpurile DEP sau REST va fi întotdeauna egal cu zero, fiind înregistrată fie o operație de depunere, fie o operație de restituire).

În cazul în care actualizarea presupune adăugare de noi articole, acestea sînt preluate dintr-un fișier pe cartele.



## CAPITOLUL VIII

### PRELUCRARI SPECIALE ÎN FORTRAN

#### 1. INSTRUCȚIUNI PENTRU ALOCAREA ZONELOR DE MEMORIE

În mod normal pentru fiecare variabilă utilizată într-un program se alocă o zonă de memorie în care se amplasează valoarea variabilei.

În scopul realizării unei economii de memorie prin alocarea de zone comune pentru mai multe variabile, se pot utiliza instrucțiunile **COMMON** și **EQUIVALENCE**.

a. Instrucțiunea **EQUIVALENCE** realizează alocarea unei zone comune de memorie pentru mai multe variabile care fac parte din aceeași unitate de program.

Instrucțiunea **EQUIVALENCE** este neexecutabilă și neetichetabilă și se scrie în program înaintea instrucțiunilor executabile, după instrucțiunile de declarare a tipului și dimensiunilor variabilelor.

*Format general*

**EQUIVALENCE** (*lista 1*), (*lista 2*), ..., (*lista N*)

unde:

*lista i* ( $i = 1, 2, \dots, N$ ) conține nume de variabile simple, nume de variabile indexate sau nume de tablouri, separate prin virgulă.

Toate variabilele din lista *i* ocupă aceeași zonă de memorie. O astfel de listă de variabile se numește listă de echivalență.

Este posibil ca variabilele din aceeași listă să nu fie de același tip. În acest caz, dacă o variabilă a cărei valoare ocupă 8 octeți de memorie este declarată echivalentă (ocupă aceeași zonă de memorie) cu o variabilă a cărei valoare ocupă 4 octeți de memorie, atunci valoarea celei de a doua variabile este aceeași cu valoarea aflată în primii 4 octeți ai primei variabile. De exemplu, în cazul secvenței de instrucțiuni:

**COMPLEX C**  
**EQUIVALENCE (C, R)**

valoarea variabilei reale **R** este aceeași cu valoarea părții reale a variabilei complexe **C**.

Dacă, într-o listă de echivalență există un element de tablou, toate celelalte elemente ale tabloului (care nu fac parte din listă) se amplasează în memorie, în funcție de poziția acestuia. De exemplu:

**DIMENSION A(5)**  
**EQUIVALENCE (B, A(2))**

În acest caz valoarea elementului **A(2)** este aceeași cu valoarea variabilei reale **B**, iar valorile celorlalte elemente ale tabloului **A** se amplasează în memorie în funcție de poziția elementului **A(2)**.



Dacă lista de echivalență conține nume de tablouri, atunci echivalența se stabilește începînd cu primul element al fiecărui tablou. În lista de echivalență pot figura tablouri de dimensiuni diferite. De exemplu:

```
DIMENSION A(2, 3), B(2, 2)
EQUIVALENCE (A, B)
```

În acest caz echivalența se stabilește astfel:

```
A(1,1) A(2,1) A(1,2) A(2,2) A(1,3) A(2,3)
B(1,1) B(2,1) B(1,2) B(2,2)
```

De asemenea, în lista de echivalență pot figura tablouri cu număr diferit de dimensiuni (număr diferit de indici). De exemplu:

```
DIMENSION A(5), B(2,2)
EQUIVALENCE (A, B)
```

Echivalența se stabilește astfel:

```
A(1)    A(2)    A(3)    A(4)    A(5)
B(1,1)  B(2,1)  B(1,2)  B(2,2)
```

Instrucțiunea **EQUIVALENCE** este utilizată în special, în două cazuri:

— În scrierea unui program pentru o aceeași variabilă s-au folosit nume diferite, situație ce se poate întîmpla atunci cînd părți ale aceluiași program sînt scrise de programatori diferiți. Pentru a nu se rescrie întregul program, se va scrie în program o instrucțiune **EQUIVALENCE** în a cărei listă vor figura toate numele date variabilei respective.

— În cursul execuției unui program nu este necesar ca anumite variabile să se afle concomitent în memoria calculatorului. Zonele de memorie în care se află valorile variabilelor a căror prezență nu mai este necesară se pot folosi pentru alte variabile utilizînd o instrucțiune **EQUIVALENCE**.

b. Instrucțiunea **COMMON** realizează alocarea unei zone comune de memorie pentru mai multe variabile care fac parte din unități de program diferite. Este o instrucțiune neexecutabilă și neetichetabilă și se scrie, în fiecare unitate de program pentru care se alocă o zonă de memorie, înaintea instrucțiunilor executabile și după instrucțiunile de declarare a tipului și dimensiunilor.

*Format general*

```
COMMON /BL1/ lista 1 /BL2/ lista 2 ... /BLN/ lista n
```

unde:

- $BL_i$  ( $i = 1, 2, 3, \dots, N$ ) este numele (eticheta) blocului comun de memorie în care sînt amplasate valorile variabilelor din lista  $i$ ;
- lista  $i$  ( $i = 1, 2, 3, \dots, N$ ) este o listă de variabile care pot fi:
- variabile simple;
- nume de tablouri urmate eventual de dimensiunile acestora.

În listele de variabile asociate blocului cu același nume (etichetă), variabilele trebuie să coincidă ca număr, ordine și tip în toate unitățile de program.

*Exemplu:*

*Program apelant*

```
.....
COMMON /BLOC/ K, X, Y(5,2)
.....
```



### *Program apelat*

```
.....  
COMMON /BLOC/L, A, B (5,2)  
.....
```

În zona de memorie numită BLOC, valoarea variabilei L este aceeași cu valoarea variabilei K, valoarea variabilei A este aceeași cu valoarea variabilei X, iar elementele tabloului B sînt aceleași cu elementele tabloului Y.

Mărimea unui bloc comun, declarat într-o unitate de program, este dată de suma unităților de memorie necesare elementelor introduse în bloc prin instrucțiunea COMMON.

Este posibil să se aloce o zonă comună de memorie, pentru unități de program diferite, în care numărul de variabile ale căror valori urmează să fie amplasate în blocul comun să nu fie același pentru fiecare unitate de program.

În acest caz zona comună de memorie se numește bloc comun general sau bloc blanc. Mărimea blocului blanc este cea care corespunde celei mai mari zone declarate.

În cazul utilizării unui bloc comun blanc, în formatul instrucțiunii COMMON nu mai apare numele (eticheta) blocului. O astfel de instrucțiune se va scrie:

a) Blocul comun blanc declarat între două blocuri etichetate este marcat prin două bare consecutive, după care urmează lista variabilelor ale căror valori sînt amplasate în blocul respectiv.

*Exemplu:*

#### *Program apelant*

```
.....  
COMMON /A/ A, B // X /Y/ K, L  
.....  
Program apelat 1  
.....  
COMMON /A/ C, D // X1, X2  
.....  
Program apelat 2  
.....  
COMMON /Y/ K1, L1  
.....
```

În programul apelant s-au declarat două blocuri comune etichetate A și Y, și un bloc comun blanc, indicat prin două bare alăturate, în care se amplasează valoarea variabilei X.

În programul apelat 1 există două zone de memorie, comune cu două dintre zonele de memorie ce apar în programul apelant zona etichetată A și zona comună generală în care prin programul apelant se amplasează valoarea variabilei X, iar prin programul apelat 1 se amplasează pe lângă valoarea variabilei X1, care este aceeași cu valoarea variabilei X, și valoarea unei noi variabile X2.

În programul apelat 2 există o singură zonă de memorie comună cu una din zonele programului apelant, numită Y.



b) Blocul comun blank declarat imediat după cuvîntul COMMON necesită doar prezența listei de variabile.

*Exemplu:*

*•Program apelant*

```
.....
COMMON X /A/ A, B/Y/ K, L
.....
```

*Program apelat*

```
.....
COMMON X1, X2 /Y/ M, N
.....
```

În programul apelant valoarea variabilei X este amplasată într-un bloc comun blank, iar prin programul apelat se amplasează, în zona comună generală, pe lângă valoarea variabilei X1, care este aceeași cu valoarea variabilei X, și valoarea variabilei X2.

Instrucțiunile EQUIVALENCE și COMMON pot fi folosite în aceeași unitate de program.

În acest caz instrucțiunea EQUIVALENCE poate avea ca efect prelungirea unui bloc comun. Această prelungire este admisă numai la sfîrșitul blocului.

De exemplu, în cazul secvenței de instrucțiuni:

```
DIMENSION A(7),
COMMON X (3), Y
EQUIVALENCE (X(1), A (1))
```

elementele tabloului A se amplasează în blocul comun blank, declarat de instrucțiunea COMMON, astfel:

```

X(1)  X(2)  X(3)  Y
A(1)  A(2)  A(3)  A(4)  A(5)  A(6)  A(7)
```

Elementele A(5), A(6) și A(7) prelungesc blocul comun blank la dreapta.

Variabilele amplasate într-un bloc comun nu pot figura ca argumente în lista de argumente a subprogramelor.

Variabilele amplasate într-un bloc comun blank pot fi inițializate prin instrucțiunea DATA.

Pentru a ilustra modul de utilizare a instrucțiunii COMMON se reia exemplul dat în cazul subprogramelor de tip SUBROUTINE, în care se calculează suma elementelor din fiecare colcană a tabloului A(10, 20).

```

C      PROGRAM APELANT
        COMMON A(10, 20)
        READ (105,1) ((A(I, J), J = 1,20), I = 1,10)
1      FORMAT (20 F4.1)
        CALL SUMA
        STOP
        END
        SUBROUTINE SUMA
C      PROGRAM APELAT
        DIMENSION VV (20)
        COMMON AA (10, 20)
        DO 3J = 1,20
            VV (J) = 0
```



```

DO 4 I = 1,10
  VV (J) = VV(J) + AA(I, J)
4  CONTINUE
3  CONTINUE
  WRITE (108, 2) (VV(J), J = 1,20)
3  FORMAT ('', 'SUMELE DE PE COLOANE
* SINT'/', 20 F6.1)
  RETURN
  END

```

Tablourile A (10,20) și AA (10,20) care apar în cele două unități de program sînt declarate într-un bloc comun blank (ocupă aceeași zonă de memorie). Prin această declarație, care figurează în fiecare unitate de program, valorile elementelor tabloului sînt transmise subprogramului de tip SUBROUTINE și, de aceea numele tabloului nu apare ca argument al subprogramului. De asemenea, afișarea rezultatelor avînd loc în subprogram, tabloul ale cărui elemente constituie rezultatele sumelor efectuate este declarat numai în această unitate de program. Din aceste motive lista de argumente a instrucțiunilor CALL și SUBROUTINE este vidă.

## 2. FACILITĂȚI SUPLIMENTARE DE REALIZARE A OPERAȚIILOR DE INTRARE/ IEȘIRE.

a. **Instrucțiuni de conversie internă.** Scrierea și citirea în diferite zone ale memoriei interne se realizează prin intermediul instrucțiunilor ENCODE și DECODE. Conversia internă a informațiilor se face conform unor specificații FORMAT.

*Instrucțiunea ENCODE* realizează transferul unor informații înregistrate într-una sau mai multe zone de memorie, într-o altă zonă de memorie, fiind astfel echivalentă cu un WRITE cu FORMAT, informația fiind însă scrisă în memorie și nu extrasă pe un suport extern.

*Instrucțiunea DECODE* transferă informații dintr-o zonă de memorie într-una sau mai multe zone de memorie, fiind astfel analogă unei instrucțiuni READ cu FORMAT, cu diferența că informațiile sînt citite din memoria internă și nu de pe suportul extern.

Este important de subliniat faptul că prin ENCODE se pot scrie zone de memorie de lungime nestandard (număr de caractere, care poate fi diferit de 4B, 8B), iar prin DECODE se pot citi informațiile înregistrate într-astfel de zone de memorie.

Într-o instrucțiune ENCODE sau DECODE se specifică:

- numărul de caractere care se transferă;
- FORMATUL conform căruia se realizează conversia;
- numele zonei de memorie în care se scrie sau din care se citește;
- numele zonelor de memorie care furnizează informațiile de scris sau care primesc informațiile citite.

*Forma generală a unei instrucțiuni ENCODE este:*

[e] ENCODE (c f, v, n) listă

unde:

- e este o etichetă, a cărei prezență este opțională;
- c — o constantă întreagă sau numele unei variabile întregi și reprezintă numărul de caractere al înregistrării interne;



- f* — eticheta unei instrucțiuni FORMAT sau numele unui tablou care conține formatul;
- v* — numele unei variabile neindexate, sau numele unui tablou sau al unui element de tablou prin care se identifică zona de memorie în care se va scrie;
- n* — numele unei variabile întregi, a cărei valoare după execuția instrucțiunii va conține numărul de caractere efectiv transferate, *n* este opțional. Prezența acestui argument permite efectuarea unui control asupra execuției instrucțiunii (se compară valoarea lui *n* cu *c*);
- listă* — o listă de intrare/ieșire.

În urma execuției unei instrucțiuni ENCODE valorile elementelor din lista de intrare/ieșire sînt convertite conform FORMAT-ului specificat prin *f*, într-unul sau mai multe șiruri de caractere (în funcție de dimensiunea listei); primele *c* caractere din șir sînt înregistrate în memorie în zona definită de *v* (1 caracter/octet).

Lungimea unui șir de caractere este definită de FORMAT.

Dacă un șir de caractere are mai puțin de *c* caractere, se completează diferența cu blankuri la dreapta, dacă depășește numărul declarat în *c*, se înregistrează numai primele *c* caractere din șir, restul nefiind neglijate.

Zona de memorie definită de *v* va fi formată dintr-una sau mai multe înregistrări cu *c* caractere. Fiecare înregistrare începe la o adresă multiplu de 4.

Instrucțiunea DECODE are forma generală:

[*t*] DECODE (*c, f, v, n*) *lista*

unde *e, c, f, v, u* și *lista* au aceeași semnificație ca în cazul instrucțiunii, ENCODE, cu precizarea că *v* identifică zona de memorie din care primele *c* caractere sînt transferate în elementele specificate în *listă*. Transferul se face conform formatului specificat de *f*.

În cazul unei citiri din memorie; numărul *c* de caractere nu trebuie să fie mai mic decît numărul de caractere din care este formată înregistrarea definită prin format. Dacă valoarea lui *c* este mai mare decît lungimea înregistrării definite prin format, caracterele suplimentare prin înregistrarea internă sînt neglijate. Adresa zonei definită de *v* este întotdeauna o adresă cuvînt-memorie.

b. *Instrucțiunea NAMELIST*. O altă facilitate a limbajului FORTRAN constă în posibilitatea transferului de date între un suport extern și memorie, fără a se utiliza în instrucțiunile de citire sau scriere liste de intrare/ieșire și fără a se utiliza pentru conversie instrucțiunea FORMAT.

Această facilitate a limbajului se folosește îndeosebi pentru punerea la punct a programelor.

Pentru realizarea unui asemenea transfer de date se asociază instrucțiunilor de citire/scriere o instrucțiune NAMELIST. Aceasta este o instrucțiune neexecutabilă, netichetabilă, care poate fi plasată oriunde într-o unitate de program, înainte de instrucțiunea de intrare/ieșire asociată.

Instrucțiunea NAMELIST are următorul format general:

NAMELIST/*X1*/*lista 1*/*X2*/*lista 2* ... /*Xn*/*lista n*

unde:

— *X1, X2, ..., Xn* sînt nume simbolice, denumite nume de NAMELIST și sînt întotdeauna încadrate de bare;



— *lista* cuprinde cel puțin un element; elementele unei liste pot fi nume de variabile neindexate sau nume de tablouri, separate prin virgulă. Liste diferite pot conține elemente comune.

Parametrii formali ai procedurilor nu pot apărea într-o astfel de listă.

Referitor la numele de NAMELIST se impun următoarele restricții:

- nu poate fi definit decât o singură dată într-o unitate de program;
- nu poate fi utilizat decât într-o instrucțiune de intrare/ieșire;
- nu poate fi utilizat ca parametru de procedură;
- trebuie să difere de orice alte nume simbolice utilizate în aceeași unitate de program.

Instrucțiunea **READ NAMELIST** are următoarea formă generală:

[*e*] **READ** (*u*, *X*)

unde:

- *u* identifică suportul de pe care se citesc datele;
- *X* este un nume de NAMELIST.

Printr-o astfel de instrucțiune se citesc de pe suportul extern valorile variabilelor care figurează într-o listă cu numele *X*.

În vederea realizării transferului și conversiei datelor acestea au, pe suportul extern, o formă specială, și anume:

- primul caracter din înregistrare este ignorat;
- al doilea caracter este &, urmează numele de NAMELIST scris fără blankuri;
- numele de NAMELIST este urmat de cel puțin un blank;
- în continuare se scriu termenii care trebuie transferați și convertiți, separați prin virgulă și un număr oarecare de blankuri;
- sfârșitul datelor de intrare se marchează prin caracterele & END.

Astfel, o instrucțiune **READ** citește o înregistrare logică formată dintr-una sau mai multe înregistrări fizice, detectarea începutului înregistrării realizându-se prin citirea caracterului &, iar a sfârșitului prin citirea caracterelor & END.

O înregistrare fizică se termină la nivelul unei virgule sau a unui blank (un termen nu poate face parte din două înregistrări fizice). Întotdeauna primul caracter al unei înregistrări fizice este ignorat.

Termenii care formează înregistrarea au următoarea formă:

$$v = C$$

unde:

- v* este un nume de variabilă neindexată, un nume de tablou sau un nume de element de tablou care apare în lista cu numele *X*;
- C* — o constantă de tip corespunzător sau un șir de constante despărțite prin virgulă, în cazul variabilelor indexate.

Când *v* este un nume de tablou, prima constantă din șir se introduce în primul element din tablou ș.a.m.d.

Când *v* este un nume de element de tablou și există un șir de constante, primul element din șir se introduce în zona afectată elementului de tablou specificat și următoarea constantă din șir se introduce în zona de memorie afectată elementului de tablou de rang imediat superior ș.a.m.d.

În cazul în care mai multe constante din șir au aceeași valoare, se poate folosi o scriere prescurtată de formă  $K * C$ , unde *K* este numărul de repetări ale constantei *C*.



*Exemplul 1:*

Să se citească, cu NAMELIST, valorile unui vector  $A$  cu 4 componente și ale variabilelor  $B$  și  $C$ . Elementele vectorului au valorile următoare:

$$\begin{aligned}A(1) &= 12.35 \\A(2) &= -405.11 \\A(3) &= 0 \\A(4) &= 0\end{aligned}$$

iar  $B = 25.02$  și  $C = -3.5$ .

Pentru aceasta se va scrie următoarea secvență de instrucțiuni:

```
DIMENSION A(4)
NAMELIST /NUME1/A, B, C
READ (105, NUME1)
```

Cartela de date va avea următorul conținut:

```
&NUME 1 A = 12.35, -405.11, 2*0., B = 25.02, C = 3.5 &END,
```

Instrucțiunea **WRITE** cu **NAMELIST** are forma generală:

[e] **WRITE** ( $u, X$ )

unde  $u$  și  $X$  au aceeași semnificație ca în cazul instrucțiunii **READ** cu **NAMELIST**.

Această instrucțiune are ca efect scrierea pe suportul extern a valorilor variabilelor declarate în prealabil într-o instrucțiune **NAMELIST**, ca aparținând unei liste cu numele  $X$ .

Datele sînt scrise într-o formă analoagă cu cea descrisă în cazul introducerii datelor cu **READ** cu **NAMELIST**.

Primul caracter este întotdeauna blank.

Valorile variabilelor sînt convertite în constante de tip corespunzător ca și cînd operația de scriere s-ar fi efectuat cu o instrucțiune **FORMAT** care conținea descriptori de forma:

$In$	pentru variabile întregi;
$1PGn.6$	pentru variabile reale;
$1PGn. 16$	pentru variabile în dublă precizie;
$L1$	pentru variabile logice.

Valoarea lui  $n$  este astfel încît zerourile nesemnificative nu apar.

O valoare complexă se reprezintă ca două numere reale separate prin virgulă și încadrate de paranteze.

Instrucțiunea **WRITE** cu **NAMELIST** oferă astfel posibilitatea editării datelor într-o formă clară și estetică, fără a fi necesară o descriere de **FORMAT**.

De asemenea, forma datelor la ieșire permite utilizarea lor ca date de intrare în cazul unei citiri cu o instrucțiune **READ** asociată aceluiași nume de **NAMELIST** (acest lucru are însemnătate atunci cînd datele de ieșire sînt perforate pe cartele sau cînd scrierea se face pe suport magnetic).

Instrucțiunea **NAMELIST** poate fi asociată și instrucțiunilor **READ** și **WRITE** pentru fișiere cu acces secvențial.

c. Subprogramele **BUFFER OUT** și **BUFFER IN**. În cazul fișierelor pe bandă magnetică, intrările și ieșirile se pot face și cu ajutorul subprogramelor standard **BUFFER OUT** și **BUFFER IN**.



Subprogramul BUFFER OUT creează un bloc pe bandă, informația fiind luată dintr-o anumită zonă de memorie și scrisă în fișier.

Subprogramul BUFFER IN citește un bloc de pe bandă, introducând informația într-o anumită zonă de memorie.

În instrucțiunea de descriere parametrii obligatorii sînt DVT:MT și RCF:B, MDO este facultativ, iar SFD interzis.

În general se utilizează aceste subprograme cînd informația care trebuie înregistrată sau extrasă din fișier este de lungime nestandard; deseori sînt folosite împreună cu instrucțiunile ENCODE și DECODE.

Apelul acestor subprograme se face prin instrucțiuni de forma

[e] CALL BUFFER IN ( $a, m, s, w, i, n$ )  
[e] CALL BUFFER OUT ( $a, m, s, w, i, n$ )

unde

—  $a$  este o constantă întreagă fără semn sau o variabilă de tip întreg prin care se specifică fișierul de intrare, respectiv de ieșire;

—  $m$  este o constantă întreagă fără semn sau o variabilă de tip întreg prin care se precizează modul de lucru ( $m = 0$  mod de lucru alfanumeric,  $m \neq 0$  mod de lucru binar);

—  $s$  este o variabilă simplă sau indexată, sau un nume de tablou prin care se desemnează adresa de început a zonei tampon din memoria internă;

—  $w$  este o constantă întreagă fără semn sau o variabilă de tip întreg și reprezintă numărul de cuvinte care sînt scrise sau citite;

—  $i$  este o variabilă de tip întreg care indică starea operației de transfer, și anume:

$i = 1$  transferul este în curs de execuție,  
 $i = 2$  transferul s-a terminat fără erori,  
 $i = 3$  s-a detectat sfîrșitul fișierului,  
 $i = 4$  transferul s-a terminat, dar cu erori;

—  $n$  este un parametru opțional reprezentat de o constantă întreagă sau o variabilă de tip întreg, la sfîrșitul transferului valoarea lui  $n$  arată numărul de cuvinte efectiv scrise sau citite.

Subprogramele BUFFER IN/OUT operează asupra unei singure înregistrări fizice din fișierul  $a$ , de aceea în cazul citirii de pe bandă, valoarea parametrului  $w$  trebuie să fie cel puțin egală cu numărul de cuvinte din înregistrarea fizică.

Operația de transfer s-a executat fără eroare dacă la sfîrșitul ei  $n = w$ .



## CAPITOLUL IX

# SEGMENTAREA PROGRAMELOR ÎN FORTRAN

În cadrul activității de programare o problemă importantă o constituie evitarea depășirii capacității memoriei disponibile pentru execuția unui program.

Se știe că în urma fazei de editare de legături programul în format IMT este depus pe discul sistem; pentru a putea fi executat el va trebui încărcat în memoria internă a calculatorului. Fiecărui program i se afectează o zonă de memorie (partiție) care trebuie să cuprindă atât instrucțiunile în cod-mașină cât și datele aferente programului.

Depășirea capacității de memorie disponibilă poate să apară în următoarele cazuri:

- programe simple, dar care au un volum mare de date de intrare/ieșire, în care caz problema se rezolvă prin organizarea datelor în fișiere;

- programe complexe, cu un număr mare de instrucțiuni, variabile și constante (evident, dacă pe lângă aceasta și volumul datelor de intrare/ieșire este mare, acestea se vor organiza de asemenea în fișiere, fără însă ca reducerea de memorie astfel obținută să fie suficientă pentru a rezolva problema).

Încercarea de execuție a unor astfel de programe complexe se poate solda cu apariția unui mesaj de eroare care specifică depășirea memoriei disponibile.

Se disting două situații diferite:

- dimensiunea partiției nu permite încărcarea programului în memorie;

- programul în format IMT depășește 64K octeți (această valoare este valabilă pentru FELIX C-256).

Pentru rezolvarea acestor situații sistemul FELIX C-256 oferă ca soluție segmentarea.

**Segmentarea** este o tehnică de programare care în esență constă în împărțirea programului în unități funcționale — numite segmente, care pot fi încărcate separat în memorie, în vederea execuției.

**Segmentul** este cea mai mică unitate funcțională care poate fi încărcată în memorie în timpul execuției unui program.

În vederea segmentării programele trebuie concepute modular, fiecare componentă realizând o anumită funcție. În ansamblu programul sursă va fi deci constituit dintr-un program principal și unul sau mai multe subprograme, înlănțuirea componentelor prin instrucțiuni de apelare urmînd să respecte logica programului.

Programul executabil va avea de asemenea o structură modulară, componentele sale fiind segmente. Segmentul care conține programul principal se numește *segment rădăcină* și se află în memorie pe toată durata de execuție a programului. Celelalte segmente, corespunzătoare subprogramelor, sînt încărcate în memorie în continuarea segmentului rădăcină, în aceeași zonă de memorie sau în zone de memorie diferite; segmentele sînt încărcate în memorie în momentul apelării subprogramelor componente.



## 1. CONSTITUIREA SEGMENTELOR. TIPURI DE SEGMENTE. ÎNLĂNȚUIREA LOR

În general segmentele sînt constituite în faza de editare de legături, din module obiect rezultate în urma compilării programului sursă.

Prin compilare se generează trei tipuri de module: *module program* (conțin de obicei instrucțiuni în limbaj mașină), *module de date* (datele utile unui program), *module de date comune* (date utile mai multor module program). Corespunzător acestor trei tipuri de module segmentele sînt:

- *segmente program*, care conțin unul sau mai multe module-program;
- *segmente de date*, care conțin fiecare cîte un modul de date;
- *segmente de date comune*, care conțin unul sau mai multe module de date comune.

Dimensiunea maximă a unui segment este de 64K. Structura unui program poate fi stabilită de utilizator, prin intermediul unor cartele de comandă specifice, sau de editorul de legături.

Modulele care alcătuiesc un segment pot proveni din compilări realizate în aceeași lucrare în care se realizează și editarea de legături, din biblioteci BT sau pot fi module obiect perforate pe cartele.

Editorul de legături înlănțuie segmentele unui program, arborescent sau liniar.

a. **Înlănțuirea arborescentă.** În cazul în care dimensiunea unui program este mai mare decît dimensiunea zonei de memorie disponibile se poate utiliza o structură de reacoperire astfel încît la un moment dat să avem în memorie o secvență din program, iar apoi să o reacoperim cu alta, atunci cînd execuția programului o cere.

Unitatea de reacoperire este segmentul program. Dacă un segment de date sau unul sau mai multe segmente comune sînt asociate unui segment program, ansamblul lor este considerat ca un singur segment program. Singurul segment care nu poate fi reacoperit este segmentul rădăcină.

Relațiile dintre segmente pot fi descrise grafic prin specificarea pentru fiecare segment a segmentelor pe care le apelează. Astfel în structura din figura IX.1 segmentul rădăcină RAD apelează segmentele S1 și S2, iar segmentul S2 apelează segmentele S3 și S4. Se obține astfel o reprezentare grafică a structurii de reacoperire reprezentare care se numește *arbore*.

Din acest motiv segmentul aflat permanent în memorie s-a numit *rădăcină*. Analogia a fost dusă mai departe, introducîndu-se și noțiunea de *ramură*. Prin ramură se înțelege ansamblul format dintr-un seg-

ment program și toate cele care îl preced.

În arbore, pornind de la rădăcină, segmentele se pot situa la diverse nivele.

Segmentele de pe un nivel oarecare sînt chemate de segmente situate mai sus în arbore și cheamă la rîndul lor alte segmente de pe un nivel inferior, conform legăturilor indicate de programator.

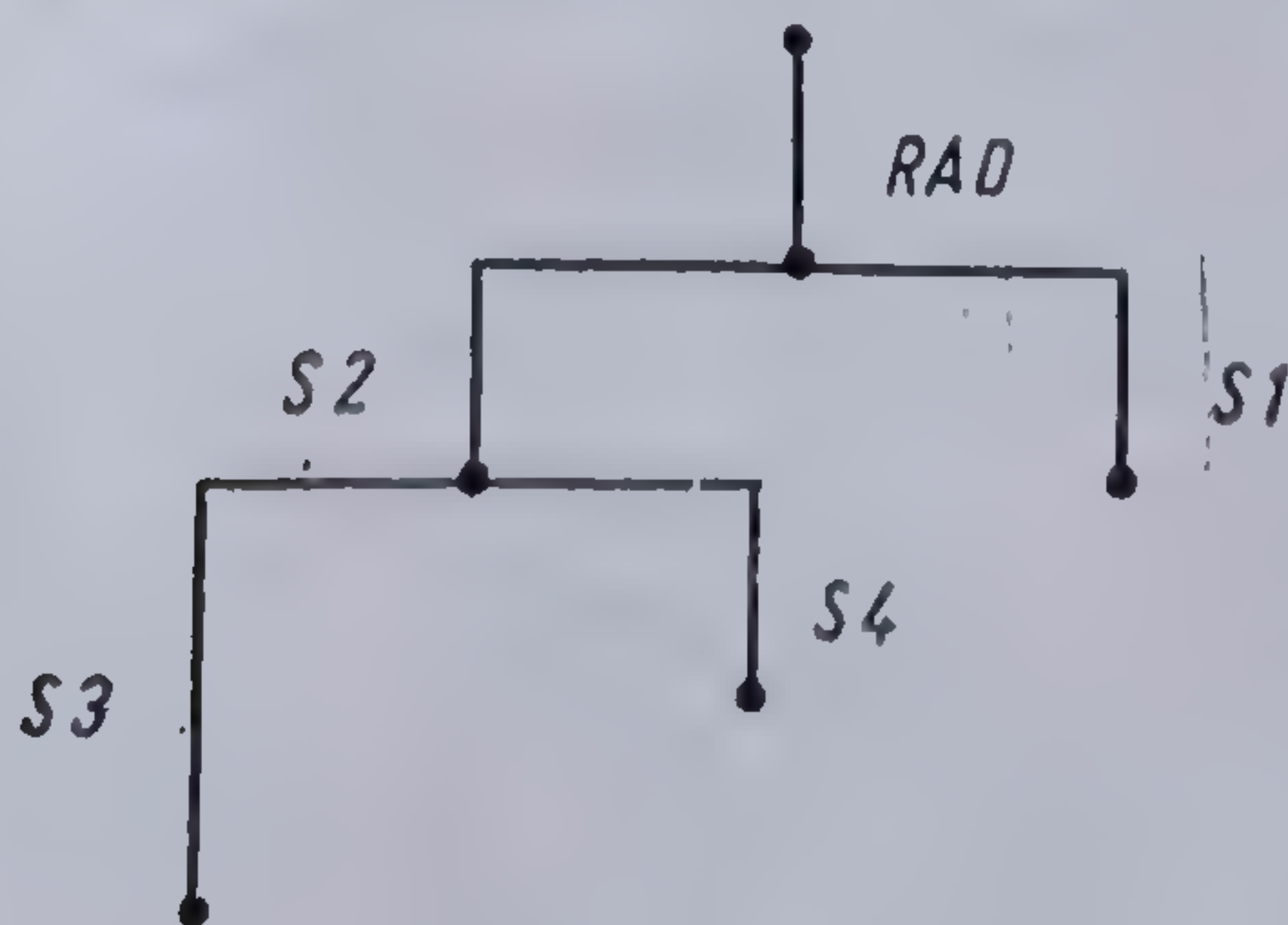


Fig. IX.1



Ultimul segment al unei ramuri este doar apelat, el neapelînd alte segmente.

b. **Înlănțuirea liniară.** În cazul înlănțuirii liniare segmentele nu se mai suprapun, ele fiind încărcate în memorie în zone distincte.

## 2. CARTELA TREE

Întrucît structura de arbore trebuie comunicată sistemului de operare, respectiv editorului de legături, se utilizează în acest scop ordinul, TREE

TREE [*arbore*]

în care:

*arbore* reprezintă descrierea structurii unui program. Această descriere este furnizată de utilizator sub forma unei expresii în care intervin nume de segmente și operatori.

*Numele segmentelor* sînt identificatori formați din  $1 \div 8$  caractere alfanumerice, primul fiind o literă. Un segment de date comune poate fi specificat fie prin numele său, fie printr-o expresie între paranteze care indică faptul că toate modulele obiect de date comune al căror nume figurează unite prin operatorul + trebuie să fie grupate într-un singur segment de date comune.

Numele segmentului de date comune rezultat din reunirea mai multor module obiect de date comune este cel al primului modul obiect de date comune specificat în expresia dintre paranteze. Editorul de legături consideră orice referire la unul din numele modulelor componente ca o referire la segmentul însuși.

*Operatorii*, în număr de patru, au următoarea semnificație:

( semnifică faptul că segmentele care urmează sînt de nivel inferior;  
) semnifică faptul că în descrierea arborelui va urma un segment de nivel superior segmentelor specificate în paranteza care s-a închis;

• , indică faptul că segmentele care urmează în descriere sînt de același nivel cu segmentele de la stînga virgulei;

+ indică faptul că segmentul care urmează în descriere este un segment de date comune atașat segmentului program precedent.

### Observații

Există un singur segment rădăcină.

Pot exista mai multe segmente de date comune de același nivel.

Încărcarea în memorie a unui segment program implică încărcarea simultană și a segmentelor de date și de date comune care li sînt atașate.

Segmentele de date sînt totdeauna implicate (nu se citează în cartela TREE).

Segmentele de date comune sînt citate explicit.

Dacă unul sau mai multe segmente de date comune sînt atașate unui segment program, acest ansamblu este considerat ca un singur segment program.

În absența cartelei TREE sau dacă expresia de descriere a arborelui este absentă, editorul de legături va da programului o structură liniară.

Cartela TREE trebuie plasată imediat înaintea cartelei LIXX.



### 3. SEGMENTAREA PROGRAMELOR FORTRAN

În absența oricărei indicații referitoare la segmentare, în urma compilării unui program FORTRAN rezultă:

- un modul obiect program pentru fiecare unitate de program (program principal sau subprogram), conținând instrucțiunile în limbaj mașină generate de compilator, datele și variabilele locale proprii modulului respectiv și, în cazul modulului obiect corespunzător programului principal, și informații cu caracter general;

- câte un modul obiect de date comune pentru fiecare bloc COMMON, etichetat sau nu.

Editorul de legături grupează într-un singur segment modulele obiect generate de compilator, împreună cu modulele obiect corespunzătoare sub-programelor necesare din biblioteca FORTRAN.

În cazul în care segmentarea este necesară, programatorul trebuie să specifice componența fiecărui segment din program, precizând pentru fiecare segment modulele componente. Se pot constitui trei tipuri de segmente: *segmente program*, *segmente de date* și *segmente de date comune*.

Un *segment program* poate fi compus din programul principal și/sau unul sau mai multe subprograme (SUBROUTINE sau FUNCTION).

Identificarea și constituirea unui segment program se face prin intermediul *ordinului*. **SEG** care are următoarea structură:

● **SEG** *nume*

-- *nume* este numele atribuit segmentului program și este format din 1—8 caractere alfanumerice, primul trebuind să fie o literă.

Ordinul SEG anunță că toate unitățile de program care îi urmează, pînă la întâlnirea unui nou ordin SEG sau a unuia din ordinele LINK sau TREE (care va fi descris ulterior), vor forma un segment program; de asemenea inițializează constituirea segmentului. Acest ordin se plasează în fața unei unități de program FORTRAN (înaintea ordinului COMPILE).

Un *segment de date* se formează prin separarea datelor de instrucțiuni, în cadrul unui modul obiect program; deci pentru o anumită unitate de program (program principal sau subprogram) va cuprinde numai datele aferente, instrucțiunile în limbaj mașină generate de compilator constituind un alt segment de tip program. Firește un segment de date este întotdeauna asociat unui segment program și este implantat în memorie odată cu acesta.

Programatorul specifică faptul că dorește constituirea unui segment de date pentru un anumit segment program plasînd în cadrul acestuia, după ordinul COMPILE, un ordin SEGDATA.

Segmentul de date asociat segmentului program va fi alcătuit din toate modulele obiect de date dintre ordinul SEG care identifică segmentul program și următorul ordin SEG, LINK sau TREE.

Un *segment de date comune* conține date accesibile mai multor segmente (date care au fost constituite într-unul sau mai multe blocuri COMMON); el este asociat unui segment program și încărcat în memorie ori de câte ori acesta este apelat.

Constituirea unui astfel de segment se specifică printr-un *ordin* **SEGMENT** sau **SEGDATA**, plasat după ordinul COMPILE din cadrul segmentului program considerat.



a. **Segmentarea programelor care depășesc dimensiunea partiției.** În cazul în care programul depășește dimensiunea partiției în care urmează să se execute, el va fi împărțit în două sau mai multe segmente program, care nu trebuie să fie prezente în memorie simultan. Segmentele vor fi pe rând încărcate în aceeași zonă de memorie, fiind preluate, în momentul apelului, de pe discul sistem. Încărcarea succesivă, în aceeași zonă de memorie, a unor segmente program se numește *reacoperire* (sau suprapunere).

Împărțirea programului în segmente program trebuie astfel realizată încât în ansamblu să se păstreze logica programului. În FORTRAN aceasta se realizează datorită faptului că un segment program nu poate fi constituit decât din unități de program. Astfel programul va fi conceput ca un program principal și unul sau mai multe subprograme, din care se vor constitui segmentele program.

Deoarece după terminarea execuției unui subprogram, adică la întâlnirea instrucțiunii RETURN, controlul trebuie transmis unității de program apelante, orice subprogram trebuie să se afle în memorie *coîncărit* cu unitatea de care a fost apelat. Întrucât această unitate poate fi la rândul său un subprogram, unitatea de program care l-a apelat pe acesta trebuie de asemenea să fie în memorie. Cele menționate fiind valabile pentru toate subprogramele rezultă că, în orice situație, în memorie trebuie să se afle programul principal. Segmentul care îl conține se numește *segment rădăcină* și rezidă în memorie pe toată durata execuției programului și, prin urmare, nu poate fi reacoperit. Acest segment dirijează încărcarea succesivă, în aceeași zonă de memorie, a celorlalte segmente program, prin intermediul instrucțiunii CALL, în cazul în care segmentul apelat este constituit din subprograme de tip SUBROUTINE (în CALL figurînd numele subprogramului sau numele punctului de intrare definit de ENTRY) sau a instrucțiunilor prin care se apelează subprogramele de tip FUNCTION.

De remarcat că în funcție de felul în care s-au grupat unitățile de program în segmente, rădăcina poate conține, pe lângă programul principal și unul sau mai multe subprograme. De asemenea, mai rezultă că după execuția fiecărui segment controlul se întoarce la segmentul rădăcină pentru a se continua execuția programului.

b. **Segmentarea programelor care nu conțin declarații COMMON.** În cazul în care un program, care nu conține declarații COMMON depășește dimensiunea partiției afectate, modulele obiect program trebuie regrupate în diferite segmente prin utilizarea ordinului SEG. Structura programului se descrie prin TREE. La alcătuirea segmentelor trebuie ținut seama de următoarele restricții:

- lungimea celei mai lungi ramuri (lungimea totală maximă a tuturor segmentelor ce pot fi prezente la un moment dat în memorie) trebuie să nu depășească lungimea partiției;

- un subprogram dintr-o ramură nu poate fi apelat dintr-un segment situat pe o ramură paralelă.

Cînd se stabilește structura segmentată a programului, în general este bine să se țină seama de frecvența de apelare a diferitelor subprograme și să se amplaseze în segmentul rădăcină modulele cele mai des apelate; acest lucru realizează o economie de timp, operația de încărcare în memorie fiind lentă.

*Exemplu:*

Se consideră un program FORTRAN alcătuit dintr-un program principal PP care apelează două subprograme SP1 și SP2. Programul principal are o lungime de 25 Kocteți, SP1 de 20 Kocteți și SP2 de 10 Kocteți.



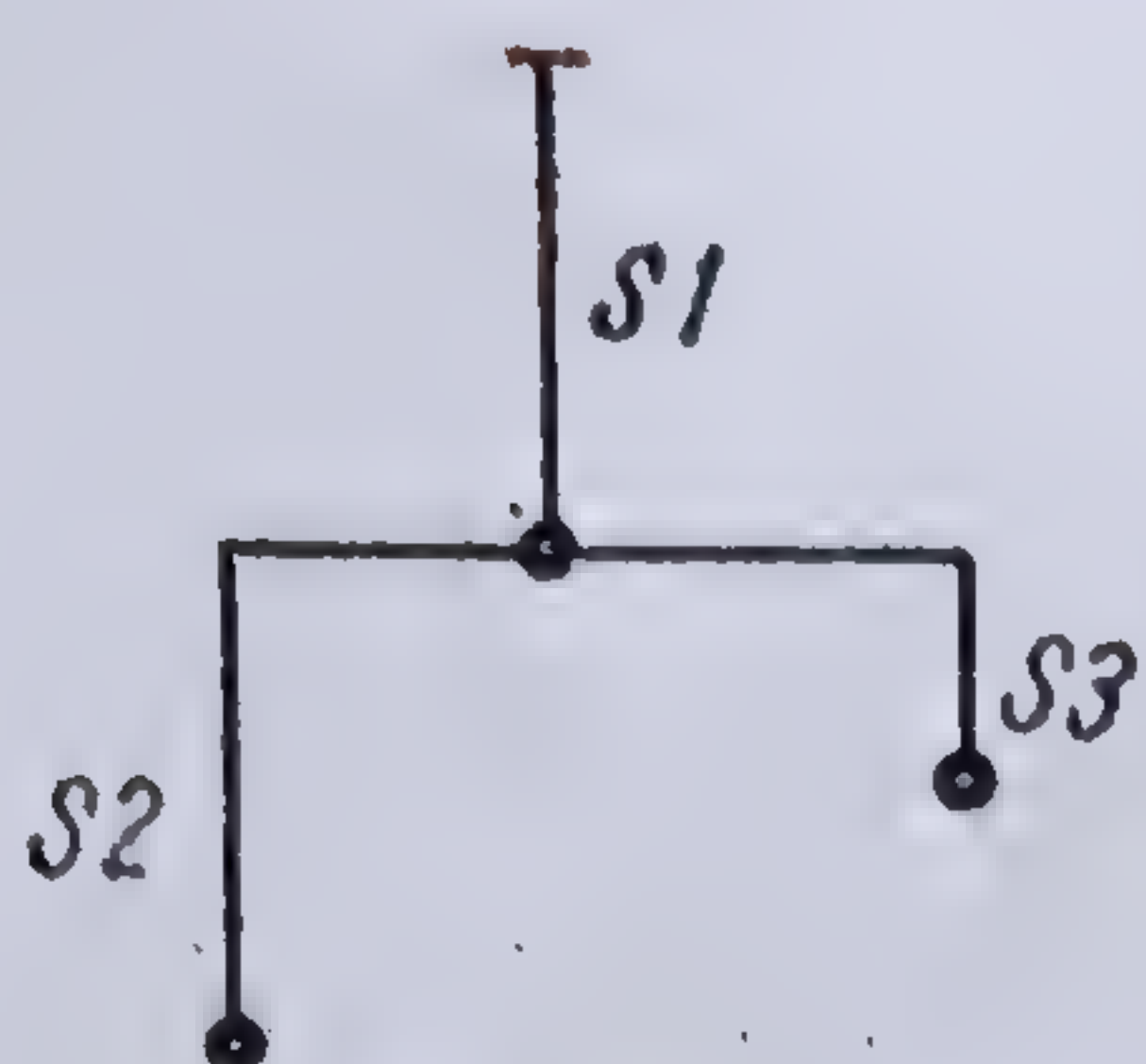


Fig. IX.2

Se presupune că dimensiunea partiției afectate programului este de 50 Kocteți. Se observă că lungimea programului este de 55 K ( $25\text{ K} + 20\text{ K} + 10\text{ K}$ ), deci programul va trebui segmentat.

Structura de reacoperire este prezentată în figura IX.2. Se observă că segmentele S2 și S3 se reacoperă reciproc când sînt apelate de segmentul rădăcina S1. Astfel la un moment dat în memoria internă a calculatorului se găsește fie segmentul S1 și S2 fie S1 și S3. Lungimea celei mai lungi ramuri (ramura S1—S2) este de 45K, deci problema este rezol-

vată. După cum s-a specificat subprogramul SP1 nu poate fi apelat din segmentul S3 și nici invers (subprogramul SP2 nu poate fi apelat din segmentul S2).

Secvența cartelelor de comandă în acest caz este următoarea:

```

.      JOB
.      SEG S1
.      COMPILE FORTRAN
.      —
.      —
.      —
.      CALL SP1
.      —
.      —
.      CALL SP2
.      —
.      —
.      END
.      SEG S2
.      COMPILE FORTRAN
.      SUBROUTINE SP1
.      —
.      —
.      —
.      END
.      SEG S3
.      COMPILE FORTRAN
.      SUBROUTINE SP2
.      —
.      —
.      —
.      END
.      TREE S1 (S2, S3)
.      LINK
.      RUN
.      date
.      EOJ
  
```

c. Segmentarea programelor care conțin declarații COMMON. Dacă programul care depășește dimensiunea partiției conține și date comune mai multor unități de program, pe lângă definirea segmentelor program trebuie



definite și unul sau mai multe segmente de date comune. Există două posibilități:

— fiecare modul obiect de date comune (blanc sau etichetat) constituie un segment distinct, purtând numele modulului (pentru modulul obiect de date comune neetichetat numele este F% BLK), sau

— mai multe module obiect de date comune se grupează într-un singur segment, purtând numele unuia din module.

În ambele cazuri programatorul trebuie să indice compilatorului FORTRAN numărul de segmente de date comune pe care dorește să le formeze. Acest lucru se realizează prin intermediul unui ordin \* SEGMENT plasat, în cazul programului principal, înaintea primei instrucțiuni FORTRAN și în cazul unui subprogram, înainte de instrucțiunea SUBROUTINE sau FUNCTION.

Ordinul \* SEGMENT are următoarea structură:

\* SEGMENT G1, G2, ..., Gn

Cărtela conține:

- caracterul \* în coloana 1;
- caracterul *blanc* în coloana 2;
- identificatorul SEGMENT în coloanele 3÷9;
- caracterul *blanc* în coloana 10;
- argumentele  $G_i$  în coloanele 11 la 71.

Fiecare argument  $G_i$  reprezintă o grupare de module obiect de date comune și este de forma:

$C$  (atunci când segmentul de date comune conține un singur modul obiect și anume pe  $C$ ), sau

$C_1 + C_2 \dots + C_p$  cu  $p \geq 2$  (atunci când segmentul grupează toate modulele  $C_1 \dots, C_p$ ). Fiecare  $C$  sau  $C_j$  ( $j = 1, p$ ) reprezintă numele unui modul obiect de date comune.

Fiecare grupare  $G_i$  face parte dintr-un segment distinct.

Într-un program FORTRAN nu se pot crea decît cel mult 4 segmente de date comune asociate unui anumit segment program (nu sînt disponibile decît 4 registre generale pentru păstrarea adreselor de bază ale segmentelor).

În cazul în care un segment grupează mai multe module obiect de date comune, numele segmentului este cel al primului modul care figurează în descrierea grupării în cartela SEGMENT.

Pentru fiecare grupare  $G_i$  se pot indica toate modulele constitutive, dar acest lucru nu este necesar, fiind suficient să se indice numele unui modul din grupare. Acest lucru este valabil însă numai pentru ordinul SEGMENT, în ordinul TREE trebuind să fie specificate toate modulele obiect de date comune care constituie o anumită grupare.

*Exemplu:*

Fie un ordin SEGMENT cu următoarea structură:

\* SEGMENT COM1 + COM2, F% BLK, COM3 + COM4 + COM5

Rezultă că cele 6 module de date comune ale unității de program vor fi grupate în trei segmente de date comune astfel:

— modulele de date comune COM1 și COM2 în primul segment de date comune;

— modulul de date comune neetichetat (F% BLK), în al doilea segment de date comune;

— modulele COM3, COM4 și COM5, în al treilea segment.



## Observații

Argumentele  $G_i$  pot fi scrise (cînd e necesar) pe mai multe cartele. În această situație cartela care se va continua conține caracterul \* în coloana 72, iar cartela de continuare conține:

- caracterul \* în coloana 1;
- caracterul *blanc* în coloanele 2+11;
- argumentele se scriu în continuare din coloana 12;
- numele unui modul COMMON nu poate figura parțial pe o cartelă și parțial pe altă cartelă; despărțirea se face la nivel de virgulă sau +, în acest caz fiind admise blaturile în cadrul descrierii.

Dacă se grupează într-un singur segment de date comune toate modulele obiect de date comune dintr-o unitate de program se poate utiliza cartela SEGMENT sub forma simplificată.

\* SEGMENT

dar gruparea trebuie neapărat indicată în cartela TREE.

### Exemplu:

Dacă o unitate de program definește blocurile comune COM1 și COM2 și un bloc comun neetichetat și se dorește gruparea celor trei module de date comune în același segment ordinul SEGMENT poate avea una din următoarele forme echivalente:

\* SEGMENT

sau

\* SEGMENT COM1 + COM2 + F% BLK

Dacă fiecare modul obiect de date comune constituie un segment distinct, numărul de blocuri COMMON ce se pot utiliza în program este cel mult egal cu 4. Gruparea mai multor module într-un segment înlătură acest dezavantaj.

d. Segmentarea programelor a căror lungime depășește 64 Kocteți. În cazul programelor a căror lungime depășește 64 Kocteți apar, de asemenea, două situații:

- programele nu conțin blocuri COMMON;
- programele conțin blocuri COMMON.

● Segmentarea programelor care nu conțin declarații COMMON. Se consideră pentru început un program alcătuit dintr-o singură unitate de program care după editarea de legături depășește 64K.

În această situație programatorul va cere ca din modulul obiect inițial să se constituie un modul obiect program, care să conțină numai instrucțiunile în limbaj mașină generate la compilare și un modul obiect care să conțină datele programului. Cel de-al doilea modul se va numi modul obiect de date. Modulului obiect de date i se atribuie numele F%MDATA, iar modulului obiect program numele F%MAIN. Editorul de legături va constitui aceste două module în segmente distincte. Astfel lungimea maximă a programului poate ajunge la 128K, fără să fie necesară vreo altă structurare (bineînțeles cu condiția ca fiecare modul în parte să nu depășească 64 K).

Cererea de constituire a două module obiect se adresează compilatorului prin intermediul unui *ordin*\* SEGDATA. Acest ordin are următoarea structură:

\* SEGDATA

Cartela conține:

- în coloana 1 caracterul \*;
- în coloana 2 caracterul *blanc*;
- în coloanele 3÷9 ordinul SEGDATA.



Cartela se plasează în program imediat după COMPILE. În acest caz deși nu există decît un segment program, este necesară cartela SEG. Astfel, programul va avea o structură liniară de tipul

```

      .      JOB
      .      SEG PRIM
      .      COMPILE FORTRAN
      *      SEGDATA
            —
            —
            —
      .      END
      .      LINK
            date
      .      RUN
            EOJ

```

Se observă că structura fiind liniară, nu se utilizează ordinul TREE.

Se consideră apoi, de exemplu, un program care a fost segmentat, dar există segmente program care depășesc 64K. Pentru aceste segmente program se vor constitui segmente de date. Constituirea segmentelor de date se specifică și în acest caz prin ordinul SEGDATA. În același segment nu se poate utiliza decît un singur ordin SEGDATA, unui segment program neputîndu-i-se asocia decît un segment de date.

Atunci cînd segmentul de date se constituie într-un subprogram, ordinul SEGDATA are o structură oarecum diferită de cea descrisă anterior prin faptul că cuprinde și specificarea numelui atribuit segmentului de date:

\* SEGDATA (*nume*)

- În coloana 10 caracterul „(“;
- *nume* — numele atribuit segmentului de date; acesta trebuie să fie diferit de orice alt nume de segment și de numele punctelor de intrare (definite prin FUNCTION, SUBROUTINE sau ENTRY);
- ultimul caracter din *nume* este urmat de caracterul „)“.

*Exemplu:*

Se consideră un program format din cinci segmente program: A, B, C, D și E. Structura de arbore este descrisă în figura IX.3. Segmentele B, C și D sînt constituite din subprogramele de tip SUBROUTINE SPB, SPC și SPD, iar segmentul E din subprogramul de tip FUNCTION ALPHA. Se presupune că B are o dimensiune mai mare de 64K și trebuie să se constituie un segment de date cu numele CICO din datele segmentului program B.

Programul va avea structura următoare:

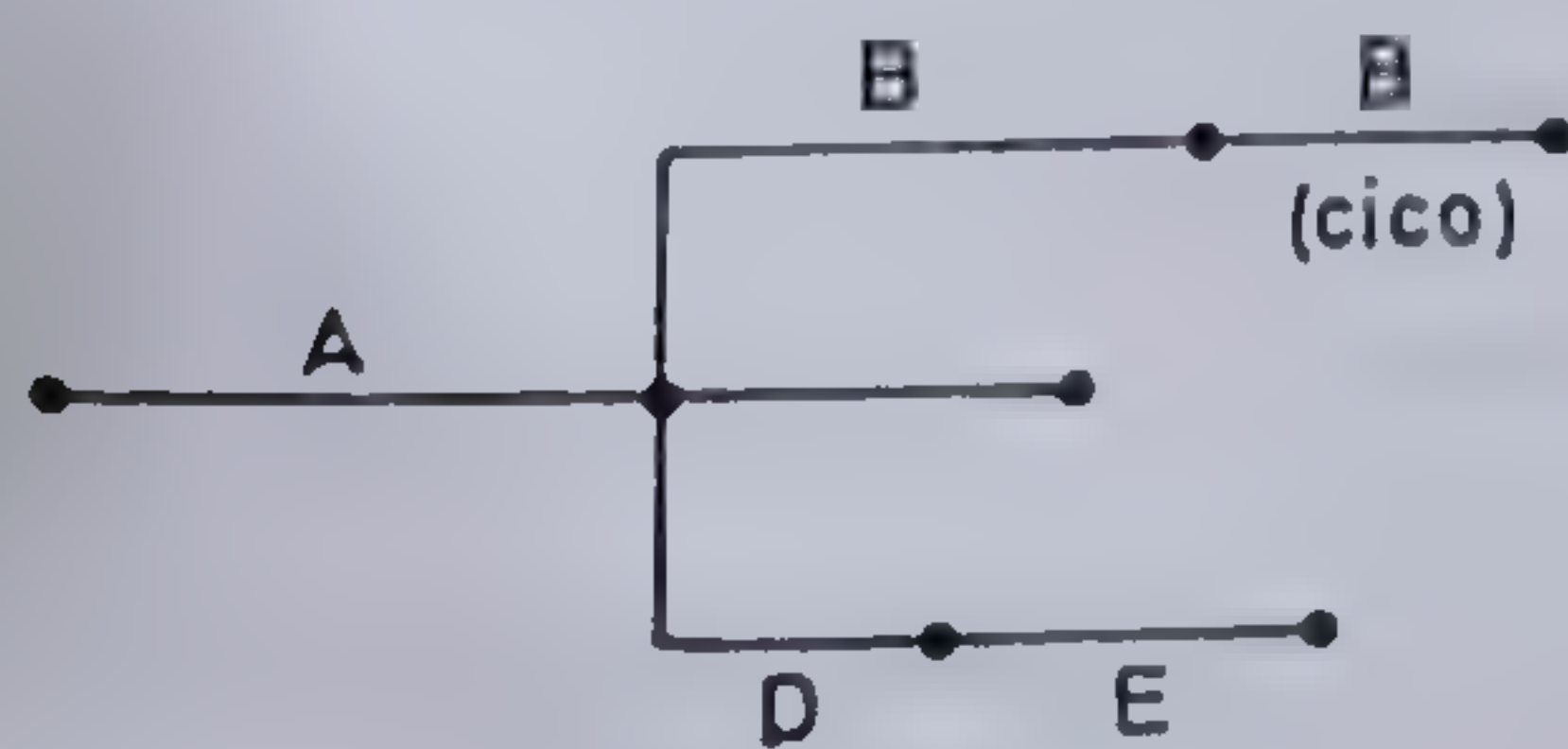


Fig. IX. 3

```

      JOB
      SEG A
      COMPILE FORTRAN
      —
      —
      —
      END
      SEG      B

```



```

                                COMPILE FORTRAN
* SEGDATA (CICO)
                                SUBROUTINE SPB
                                ---
                                ---
                                END
                                SEG C
                                COMPILE FORTRAN
                                SUBROUTINE SPC
                                ---
                                ---
                                END
                                SEG D
                                COMPILE FORTRAN
                                SUBROUTINE SPD
                                ---
                                ---
                                END
                                SEG E
                                COMPILE FORTRAN
                                FUNCTION ALPHA
                                ---
                                ---
                                END
                                TREE A (B, C, D (E))
                                LINK
                                RUN
                                date
                                EOJ

```

Segmentul de date CICO este asociat numai segmentului program *B* și este încărcat în memorie atunci când segmentul *B* este apelat (la întâlnirea instrucțiunii CALL SPB).

● **Segmentarea programelor care conțin declarații COMMON.** Se consideră un program care conține un program principal și mai multe subprograme care folosesc date în comun și care a fost constituit în segmente program și segmente de date comune, pentru a nu depăși dimensiunea partiției. Se poate întâmpla ca anumite segmente să aibă în continuare o dimensiune prea mare, în sensul că depășesc lungimea maximă admisibilă pentru un segment (64K). Pentru aceste segmente se vor constitui și segmente de date. Dacă aceste segmente program conțin și declarații COMMON, pentru constituirea segmentelor de date comune se va folosi tot ordinul \* SEGDATA, care de această dată va avea un dublu rol: de despărțire a modulului obiect într-un modul obiect de date și un modul obiect program, și de a da indicații asupra modului de construire a segmentelor de date comune (același rol cu ordinul \* SEGMENT).

În această situație ordinul \* SEGDATA are una din următoarele forme:

```
* SEGDATA G1, G2, ..., Gn
```



în cazul unui program principal și

\* SEGDATA (*nume*)  $G_1, G_2, \dots, G_n$   
în cazul unui subprogram (SUBROUTINE sau FUNCTION).

—  $G_i$  are aceeași semnificație ca în cazul ordinului

\* SEGMENT

— *nume* este numele segmentului de date.

Deoarece pentru păstrarea adreselor de bază ale segmentului de date și segmentelor de date comune asociate unui segment program nu sînt disponibile decît patru registre generale, rezultă că unui segment program care are asociat un segment de date nu i se pot asocia decît maximum 3 segmente de date comune. Astfel numărul grupărilor  $G_i$  dintr-o cartelă \* SEGDATA este maximum egal cu trei.

Observații

În cadrul aceluiasi program anumite unități de program pot avea asociat cîte un segment de date, iar altele nu.

Există situații în care rezultatele obținute în urma execuției unui program diferă în funcție de faptul că programul a fost segmentat sau nu.

Fie, de exemplu, structura de reacoperire din figura IX. 4, în care S1 reprezintă un program principal, iar S2 și S3 subprograme.

Dacă ordinea de apel a subprogramelor S2 și S3 este astfel încît în memorie există succesiv ramurile:

S1 — S2  
S1 — S3  
S1 — S2  
S1 — S2

în cazul primului apel al lui S2 se vor inițializa anumite variabile. Aceste variabile vor fi reinițializate la cel de al doilea apel al lui S2, ele nemaiavînd valorile rezultate în urma execuției subprogramului S2. La cel de-al treilea apel al lui S2 nu vor mai fi reinițializate segmentul S2 fiind deja încărcat în memorie.

De această observație trebuie ținut seama și în cazul apelurilor succesive de segmente program care au asociate segmente de date sau segmente de date comune.

Dacă un segment program conține unul din ordinele \* SEGMENT sau \* SEGDATA și unul sau mai multe ordine \* DEFINE FILE, acestea se plasează înaintea primei instrucțiuni FORTRAN, ordinea lor fiind indiferentă.

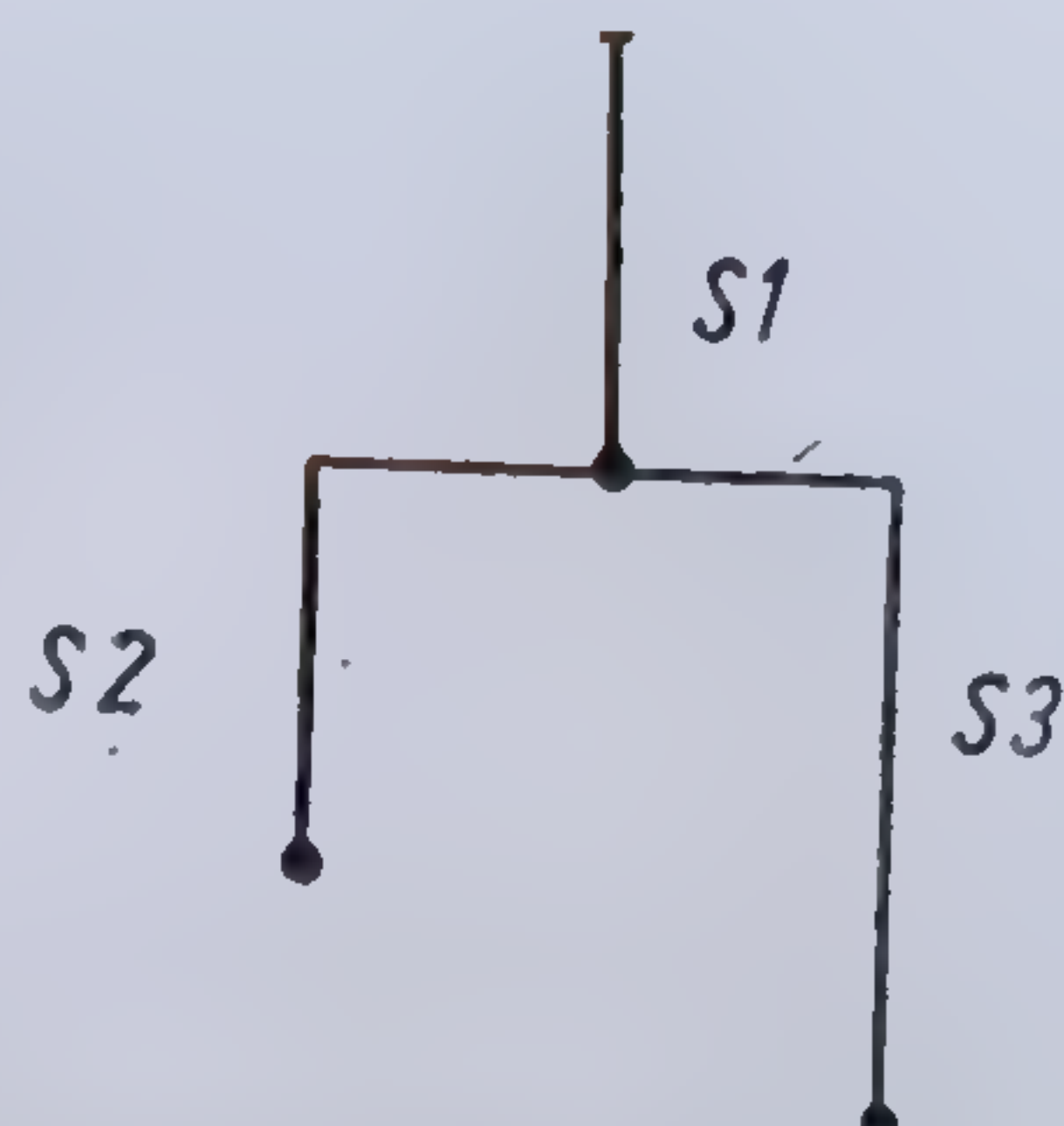


Fig. IX.4



## CAPITOLUL X

### PARTICULARITĂȚI ALE LIMBAJULUI FORTRAN ÎN CAZUL MINICALCULATOARELOR FELIX

Familia de minicalculatoare românești este înzestrată cu un sistem de operare multifuncțional, care poate acoperi o gamă largă de aplicații. Acest sistem de operare se numește AMS (*Adaptable Multifunctional System*). Sistemul AMS permite execuția programelor de timp real, precum și dezvoltarea și execuția de programe, fie prin prelucrarea în loturi, fie interactiv, de la un singur terminal, fie în timp partajat, simultan de la mai multe terminale.

AMS este un sistem de operare orientat pe fișiere. Astfel, programul sursă va sta la baza unui fișier sursă; în urma compilării va rezulta un fișier obiect, iar în urma editării de legături, un fișier imagine executabilă. Datele de intrare sau de ieșire sînt de asemenea, organizate în fișiere, cu excepția cazului în care sînt introduse sau extrase direct de la terminal.

Fișierele pot fi create și prelucrate pe discuri sau benzi magnetice, accesul la informații fiind direct sau secvențial.

Sistemul de operare AMS permite prelucrarea de programe în FORTRAN IV, limbajul implementat în acest sistem prezentînd, față de limbajul FORTRAN pentru FELIX C-256, următoarele principale particularități:

— alfabetul admite și caracterele speciale:

„ ghilimele  
: două puncte  
\$ dolar  
; punct și virgulă

TAB caracter de tabulare;

— identificatorii au numai primele șase caractere semnificative;

— constantele hexazecimale sînt înlocuite de constante octale; o constantă octală este de forma:

„ n

— n fiind un șir de cifre octale și  $0 \leq n \leq 177777$ ; (8)

— constantele alfanumerice sînt formate din caractere ASCII;

— sînt admise și constante RADIX-50, de forma:  $nRc_1, \dots, c_n$  cu  $n \leq 12$ . Aceste constante nu pot fi utilizate decît în instrucțiuni DATA;

— la sfîrșitul fiecărei linii sursă se pot introduce comentarii, precedate de semnul ! ;

— liniile de compilare condiționată au caracterul D în prima coloană; opțiunea de compilare condiționată se specifică prin introducerea comutatorului / DE în linia de comandă a compilării;

— s-au introdus și operații logici .XOR. (sau exclusiv) și .EQV. (echivalența logică);



— operațiile logice se pot efectua și cu constante sau variabile de tip întreg; în acest caz operația logică se efectuează la nivelul biților și rezultatul este de tip întreg. Fie de exemplu următoarea secvență:

I = 4  
J = 5  
K1 = I.AND.6  
K2 = J.OR.6

În urma execuției acestor instrucțiuni, K1 va primi valoarea 4 ( $100_{(2)}$ ), iar K2, 7 ( $111_{(2)}$ );

— în listele de intrare-ieșire și în instrucțiunile DO și GO TO calculat se pot utiliza expresii de format general;

— admite noi instrucțiuni de intrare-ieșire, orientate pe periferie. Acestea sînt: ACCEPT, analoagă unui READ cu FORMAT, datele fiind citite de la consola sistemului și TYPE, analoagă unui WRITE cu FORMAT, datele fiind, în acest caz, afișate la consola sistemului. Formatul instrucțiunilor este:

e ACCEPT f, listă de variabile  
e TYPE f [,listă de variabile]

— nu se acceptă instrucțiunile USED FILE și BUFFER IN/OUT;

— fișierele exploatate în acces secvențial nu sînt descrise prin instrucțiunea DEFINE FILE, aceasta fiind rezervată fișierelor în acces direct;

— operațiile de intrare-ieșire în acces direct se realizează cu ajutorul instrucțiunilor READ și WRITE în acces direct, fără format. Fișierele exploatate în acces direct sînt descrise prin instrucțiuni DEFINE FILE de forma:

DEFINE FILE u (m, n, U, v) [,u (m, n, U, v) ...]

unde:

u este o constantă sau variabilă întregă, care specifică numărul unității logice asociată fișierului;

m este o constantă sau variabilă de tip întreg, care specifică numărul de articole din fișier;

n este o constantă sau variabilă de tip întreg, precizînd lungimea în cuvinte a articolului;

U indică operație de intrare-ieșire fără format (nu se acceptă decît acest caracter);

v este variabila asociată și are aceeași funcție ca în cazul fișierelor în acces direct din cadrul sistemului FELIX C-256.

De exemplu, instrucțiunea:

DEFINE FILE 3(2 000, 64, U, IV)

asociază unitatea logică 3 unui fișier în acces direct cu 2000 de articole, fiecare articol avînd lungimea de 64 de cuvinte. În fișier articolele sînt numerotate de la 1 la 2 000. După fiecare operație de intrare-ieșire IV va conține numărul articolului care urmează celui prelucrat.

Instrucțiunea DEFINE FILE trebuie să fie plasată în program înaintea instrucțiunilor READ sau WRITE care exploatează fișierul specificat.

— nu se admit intrări și reveniri multiple dintr-un subprogram (instrucțiunile ENTRY și RETURN; nu sînt acceptate);

— este permis accesul la directivele sistemului;

— programele FORTRAN pot apela direct rutine scrise în limbaj de asamblare;

— se pot realiza aplicații complexe de timp real.



### Exemplu

În continuare se prezintă un program FORTRAN simplu, care calculează suma componentelor unui vector. Programul a fost dezvoltat și executat interactiv de la terminal (TTO — consola sistemului).

```
>EDI DK3:TASK1.FTN  
[CREATING NEW FILE]  
INPUT
```

```
C TEST
```

```
      DIMENSION IVECT(10)  
      TYPE 10  
10    FORMAT('  SUMA COMPONENTELOR UNUI VECTOR  
      ACCEPT 1,IVECT  
1     FORMAT(10I3)  
      TYPE 2,IVECT  
2     FORMAT('  VECT=(',9(I3,', '),I3,')')  
      IS=0  
      DO 3 I=1,10  
          IS=IS+IVECT(I)  
3     CONTINUE  
      TYPE 4,IS  
4     FORMAT('  SUMA= ',I6)  
      STOP  
      END
```

```
AZ
```

```
>RUN $FOR
```

```
FOR>DK3:TASK1.OBJ=DK3:TASK1.FTN
```

```
FOR>AZ
```

```
>LNK DK3:TASK1.EXE=DK3:TASK1.OBJ
```

```
>RUN DK3:TASK1.EXE
```

```
  SUMA COMPONENTELOR UNUI VECTOR  
1,2,344,56,23,34,34,56,57,10  
  VECT=(  1,  2,344, 56, 23, 34, 34, 56, 57, 10)  
  SUMA =   617
```

### PROGRAMUL X.1

Pentru crearea fișierului sursă s-a utilizat un program utilitar. Editorul de texte — EDI. Prin EDI DK3: TASK1. FTN s-a apelat utilitarul și s-a specificat fișierul sursă (fișierul sursă este înregistrat pe discul DK3:, are numele TASK1 și este de tip. FTN, adică în FORTRAN). Semnul > care precede comanda arată că se pot introduce comenzi de la terminal. Programul EDI a răspuns că este gata să creeze fișierul sursă solicitat, afișând la terminal mesajul:

```
[CREATING NEW FILE]  
INPUT
```

În continuare s-au introdus de la terminal liniile sursă. După introducerea liniei END s-a eliberat editorul de texte printr-o comandă care produce afi-



șarea caracterului ^Z. În acest moment s-a terminat operația de creare a fișierului sursă, care este de acum înregistrat pe disc.

Prin `RUN $FOR` s-a apelat compilatorul FORTRAN. Acesta se prezintă prin `FOR >` și se introduce linia de comandă care în dreapta semnului `=` are descrierea fișierului sursă, iar în stînga, descrierea fișierului obiect.

Compilatorul se prezintă în continuare în vederea introducerii de noi comenzi. Cînd nu este cazul, se eliberează.

Se apelează apoi editorul de legături — LNK și se cere editarea legăturilor pentru fișierul obiect `DK3: TASK1.OBJ` în vederea obținerii fișierului imagine executabilă (de tip `.EXE`), cu numele `TASK1`, pe discul `DK3: .` Se cere apoi execuția programului.

Începe execuția programului și se afișează la terminal mesajul din instrucțiunea `FORMAT` cu eticheta 10 (în urma execuției instrucțiunii `TYPE 10`). Execuția programului continuă cu instrucțiunea `ACCEPT`, și de la consola sistemului sînt introduse datele (valorile celor 10 componente ale vectorului, despărțite de virgulă). Programul preia datele de intrare, continuă execuția și afișează la consolă rezultatele (în urma execuției instrucțiunilor `TYPE 2`, `1VECT` și `TYPE 4, IS`).



## PARTEA A DOUA

---

### PROGRAMAREA ÎN LIMBAJUL COBOL

#### CAPITOLUL XI

#### NOȚIUNI INTRODUCTIVE

##### 1. ISTORIC

Introducerea într-o măsură tot mai mare a sistemelor de prelucrare automată a datelor în domeniile: financiar, contabil, comercial etc., domenii a căror caracteristică este volumul mare de date prelucrate prin operații relativ simple, a impus necesitatea creării unui limbaj de programare adecvat acestor prelucrări, accesibil și care să nu fie dependent de un anumit tip de calculator.

Problema creării acestui limbaj a fost discutată pentru prima dată la conferința organizată, în mai 1959, la Washington. Lucrările conferinței au stabilit că realizarea limbajului este necesară și totodată posibilă. Comitetul conferinței — CODASYL (Conference on DATA SYstems Languages) — numește un comitet pentru coordonarea lucrărilor de elaborare a limbajului. Acest comitet s-a întrunit în iunie 1959, stabilind un grup de lucru. Sarcina grupului de lucru a constat în examinarea limbajelor existente (FLOW-MATIC, AIMACO, CONTRAN etc.) și stabilirea necesității unui nou limbaj. Concluzia grupului de lucru a fost: elaborarea unui nou limbaj, pentru care se propune numele COBOL (Common Business Oriented Language).

Pe baza specificațiilor elaborate în noiembrie 1959 este realizată prima versiune a limbajului COBOL, care a fost publicată în aprilie 1960. În același an, a fost creat un comitet pentru dezvoltarea limbajului care, pe baza observațiilor și recomandărilor referitoare la prima versiune, elaborează a doua versiune, publicată în anul 1961 sub numele COBOL-61. Versiunea următoare, publicată în 1962 sub numele COBOL-61 *Extended*, prezintă, în plus, facilitățile: editare de rapoarte și sortare de fișiere. Versiunile ulterioare — care au adus clarificări, modificări și extensii versiunilor anterioare — au fost publicate în anii 1965, 1969, 1970 etc, evoluția limbajului fiind facilitată de publicația *COBOL Journal of Development*.

Activitatea de standardizare a limbajului a fost inițiată în 1963, de către un comitet al ASA (American Standard Association), astăzi denumit ANSI (American National Standards Institute). Prima versiune standardizată — COBOL ANS (American National Standard) a fost publicată în 1968.



## 2. CARACTERISTICI

Fiind foarte apropiat de limbajul natural (limba engleză), limbajul COBOL este accesibil unui cerc larg de utilizatori.

Prin clauzele de descriere și instrucțiunile de care dispune, limbajul COBOL permite definirea organizărilor clasice de fișiere (secvențiale, secvențiale-indexate, selective), implicate în rezolvarea problemelor de gestiune, precum și a operațiilor prin care se realizează prelucrarea acestor fișiere (creare, consultare, actualizare, sortare). De asemenea, comparativ cu alte limbaje de programare, limbajul COBOL prezintă facilități suplimentare pentru definirea și realizarea de rapoarte.

Totuși, prin setul de instrucțiuni aritmetice și de control, prin formatele diverse de reprezentare a datelor pe care le admite, prin posibilitatea definirii și prelucrării tablourilor, limbajul COBOL poate fi utilizat și în rezolvarea problemelor din alte domenii.

Termenul „common” sugerează faptul că programele COBOL sînt, în general, compatibile, putînd fi compilate pe orice tip de calculator care are implementat un compilator COBOL.

În cadrul acestui manual este prezentată versiunea limbajului COBOL-ANS 1968, implementată la calculatoarele de tip FELIX.

## 3. METALIMBAJUL PENTRU DESCRIEREA SINTAXEI

Metalimbajul reprezintă un ansamblu de simboluri și reguli cu ajutorul cărora se alcătuiesc construcțiile ce descriu sintaxa limbajului COBOL.

● Simbolurile metalimbajului sînt: *parantezele drepte, acoladele și punctele de suspensie*. Semnificația acestora este următoarea:

- parantezele drepte încadrează o construcție COBOL opțională;
- acoladele încadrează unul sau mai multe elemente ale unei construcții COBOL, indicînd faptul că un singur element poate fi utilizat în acel loc al construcției;
- punctele de suspensie, care pot apărea în interiorul sau la sfîrșitul construcției indică repetarea de un număr de ori a unui element al construcției sau a construcției însăși.

● Regulile de interpretare a construcțiilor COBOL, prezentate prin intermediul metalimbajului, sînt următoarele:

- cuvintele scrise cu litere mari sînt *cuvinte rezervate*;
- cuvintele scrise cu litere mici reprezintă *elemente ale construcțiilor ce vor fi precizate de utilizator* (prin unități structurale ale limbajului, cum ar fi: identificatori, constante etc.) în momentul inserării acestora în programe;
- cuvintele tipărite cu aldine (îngroșate ex. **BLOC**) sînt *cuvinte cheie*, care trebuie să apară obligatoriu în cadrul construcției, iar cuvintele tipărite cu drepte (ex. **CONTAINS**) sînt *cuvinte opționale*.

*Cuvintele rezervate reprezintă succesiuni de caractere care au o semnificație bine definită în cadrul limbajului și care nu pot fi utilizate pentru referirea fișierelor, articolelor, datelor, instrucțiunilor etc.*







## CAPITOLUL XII

# ELEMENTELE DE BAZĂ ALE LIMBAJULUI COBOL

## 1. STRUCTURA PROGRAMELOR COBOL

Elementele de bază ale limbajului și structura programelor COBOL vor fi prezentate prin intermediul programului XII.1, în care sînt descrise două fișiere numite:

— FIȘIER-INTRARE (fișier pe cartele), ale cărui articole conțin date referitoare la elevii unui liceu și anume: clasa, numele, adresa și media anuală:

— FIȘIER-IEȘIRE (fișier la imprimantă), ale cărui articole conțin aceleași date.

Pentru a simplifica referirea în program a articolelor și a datelor, acestora li s-au asociat nume (identificatori) diferite:

ART-I, CLASA-I, NUME-I, ADRESA-I, MEDIE-I, și, respectiv, ART-E, CLASA-E, NUME-E, ADRESA-E și MEDIE-E. Algoritmul descris de program indică citirea succesivă a articolelor fișierului FIȘIER-INTRARE și tipărirea la imprimantă, după citirea fiecărui articol, a unui rînd ce conține datele unui elev. Prelucrarea ia sfîrșit în momentul în care este detectată marca de sfîrșit de fișier a fișierului FIȘIER-INTRARE. În cazul în care fișierul FIȘIER-INTRARE nu conține niciun articol, prelucrarea constă în tipărirea unui mesaj la imprimantă (vezi schema din figura XII.1).

Programele COBOL au o structură arborescentă formată din construcții standard dispuse pe mai multe niveluri. La primul nivel sînt situate *diviziunile*, care reprezintă părți ale programului ce grupează informații de un anumit tip. În cadrul unei diviziuni, aceste informații sînt regrupate în *secțiuni*, iar în cadrul secțiunilor, în *paragrafe (rubrici)*. Diviziunile sînt în număr de patru și apar obligatoriu în ordinea următoare: IDENTIFICATION DIVISION, ENVIRONMENT DIVISION, DATA DIVISION, PROCEDURE DIVISION.

*Titlul unei diviziuni, format din două cuvinte rezervele urmate de un punct, trebuie să figureze singur pe o linie a formularului și se scrie din coloana 8.*

În programul XII.1, pentru citirea cu ușurință a acestuia, liniile ce conțin titlurile diviziunilor sînt urmate de cîte o linie-comentariu. În cadrul unui program COBOL o linie este considerată comentariu dacă, în coloana 7, conține caracterul \*. O linie comentariu poate conține orice combinație de caractere.

a. **IDENTIFICATION DIVISION** (diviziunea de identificare). Este formată numai din paragrafe și grupează informațiile necesare identificării programului. Formatul general al acestei diviziuni este următorul:

**{ IDENTIFICATION  
  ID }** DIVISION.

**PROGRAM-ID.** *nume program.*

**[AUTHOR.** *nume autor.*]

**[INSTALLATION.** *centrul de calcul și sistemul de calcul.*]



[DATE-WRITTEN. data scrierii programului.]  
 [DATE-COMPILED. data compilării programului.]  
 [SECURITY. destinația programului.]  
 [REMARKS. funcția programului.]

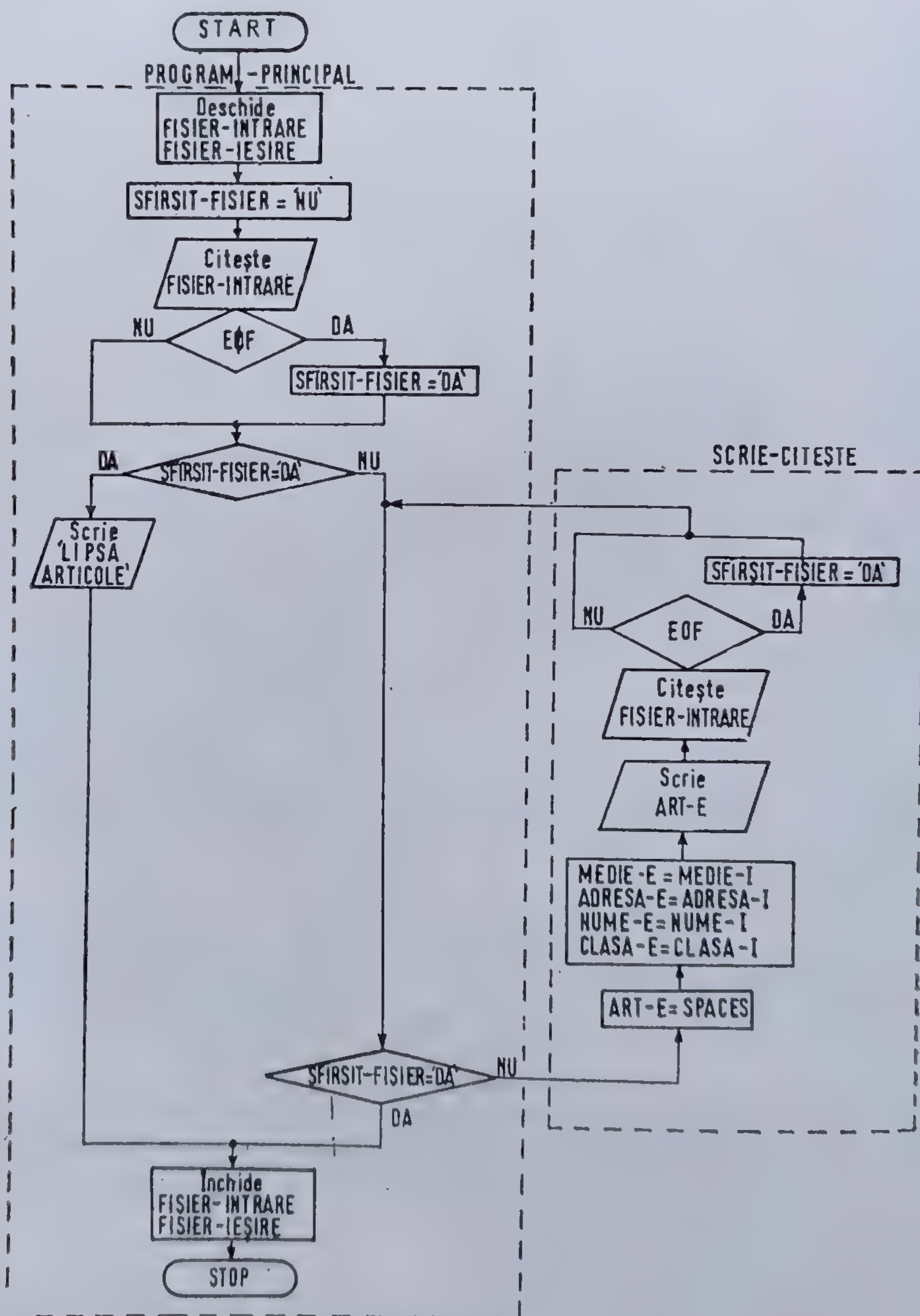


Fig. XII.1



Singurul paragraf obligatoriu este paragraful PROGRAM-ID care indică numele programului; *numele programului este un identificador format din maximum opt caractere*. Celelalte paragrafe conțin informații a căror semnificație este indicată chiar de titlul paragrafului; aceste informații sînt însă considerate de compilator comentarii.

*În general, un paragraf conține una sau mai multe fraze; fraza reprezintă o unitate structurală care se termină printr-un punct urmat de un spațiu. Fiecare paragraf este identificat printr-un nume (care este un cuvînt rezervat), care se scrie pe formular din coloana 8 și este urmat de un punct și cel puțin un spațiu. Frazele ce compun paragrafele urmează numelor acestora, însă pot fi scrise numai în coloanele 12—72.*

În cazul diviziunii IDENTIFICAȚION, cu excepția paragrafului PROGRAM-ID, toate paragrafele sînt paragrafe-comentariu. Frazele ce compun paragrafele-comentariu reprezintă succesiuni de caractere COBOL. *Caracterele permise în limbajul COBOL sînt de trei tipuri: numerice, alfabetice și speciale (tabelul XII.1). În limbajul COBOL se numește caracter alfanumeric, un caracter oarecare (numeric, alfabetic sau special).*

TABELUL XII.1

Caractere de bază ale limbajului COBOL

TIP	CARACTERE
numerice	0, 1, 2, 3, 4, 5, 6, 7, 8 și 9
alfabetice	A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z și spațiul
speciale	+ - * / = \$ , . ; ' ( ) < și >
alfanumerice	orice caracter numeric, alfabetic sau special

b. **ENVIRONMENT DIVISION** (diviziunea de echipament). Poate conține două secțiuni: CONFIGURATION SECTION și INPUT-OUTPUT SECTION; informațiile grupate în această diviziune sînt specifice sistemului de calcul și versiunii limbajului utilizat. *Titlul unei secțiuni — format din numele secțiunii cuvîntul SECTION și un punct urmat de spațiu — se scrie singur pe o linie a formularului, începînd din coloana 8. Ambele secțiuni ale diviziunii ENVIRONMENT sînt formate din paragrafe. Frazele acestor paragrafe pot conține comentarii, clauze pentru descrierea fișierelor etc.*

Paragrafele secțiunii CONFIGURATION sînt:

- **SOURCE-COMPUTER**, care indică sistemul de calcul care realizează compilarea programului (informație considerată de compilator comentariu);
- **OBJECT-COMPUTER**, care indică sistemul de calcul care realizează execuția programului (informație considerată de compilator comentariu);
- **SPECIAL-NAMES**, care precizează caracterele care pun în evidență, la editarea (tipărirea) valorilor datelor, marca zecimală și simbolul monetar.



Paragratele secțiunii INPUT-OUTPUT sînt:

— *FILE-CONTROL*, care precizează caracteristicile fișierelor utilizate în program;

— *I-O-CONTROL*, care indică tehnicile particulare de prelucrare a fișierelor.

Formatul general al diviziunii ENVIRONMENT este următorul:

**ENVIRONMENT DIVISION.**

**CONFIGURATION SECTION.**

[**SOURCE-COMPUTER.** paragraf comentariu.]

[**OBJECT-COMPUTER.** paragraf comentariu.]

[**SPECIAL-NAMES.** clauze.]

**INPUT-OUTPUT SECTION.**

**FILE-CONTROL.**

$$\left\{ \text{SELECT nume-fișier ASSIGN TO} \left\{ \begin{array}{l} \text{SYSIN} \\ \text{SYSOUT} \\ \text{SYSPUNCH} \end{array} \right\} \right\} \dots$$

[**I-O-CONTROL.** clauze.]]

În paragraful *FILE-CONTROL* există cite o frază *SELECT* pentru fiecare fișier utilizat în program. În fraza *SELECT*, prin clauza *SELECT* se indică numele (*nume-fișier*) asociat fișierului, iar prin clauza *ASSIGN*, *identificatorul de exploatare (idex)* care, dacă fișierului îi este asociată o unitate standard a sistemului, este precizat prin unul dintre cuvintele rezervate *SYSIN*, *SYSOUT*, *SYSPUNCH*. Semnificația acestora este următoarea:

*SYSIN* — unitatea de intrare a sistemului;  
*SYSOUT* — unitatea de ieșire a sistemului;  
*SYSPUNCH* — unitatea de perforare a sistemului.

Numele asociat fișierului prin clauza *SELECT*, reprezintă un identificator. În limbajul *COBOL*, *identificatorii*, *numiți și cuvinte utilizator*, reprezintă succesiuni de caractere alfabeticе (exceptînd caracterul „spațiu”), numerice și caracterul — (liniuță), fiind utilizați pentru referirea programelor, fișierelor, articolelor, datelor, instrucțiunilor etc. (în formatele generale ale construcțiilor *COBOL*, ei vor fi indicați prin notațiile *nume-program*, *nume-fișier*, *nume-articol*, *nume-dată* etc.). Ca regulă generală, *identificatorii* au o lungime maximă de 30 de caractere, conțin cel puțin o literă, iar *liniuța*, dacă este utilizată, nu poate fi primul sau ultimul caracter.

Exemple de identificatori:

*FIȘIER-BANDA*; *MODUL-10*; *COD-MATERIAL*; *NUME-ELEV*

c. **DATA DIVISION** (diviziunea de date). Conține descrierea fișierelor și a datelor prelucrate prin execuția programului și poate fi structurată în patru secțiuni:

— *FILE SECTION*, care figurează în programele care utilizează fișiere și conține descrierea acestora și a articolelor ce le sînt asociate;

— *WORKING-STORAGE SECTION*, care apare în program numai dacă în prelucrare sînt implicate date (constante, rezultate intermediare, indici etc.) care nu aparțin articolelor fișierelor și conține descrierea acestora;

— *LINKAGE SECTION*, care apare numai în cadrul subprogramelor *COBOL* și conține descrierea parametrilor formali;

— *REPORT SECTION*, care figurează numai în programele care editează rapoarte, prin intermediul editorului de rapoarte, și conține descrierea acestora.



Descrierea fișierelor, rapoartelor, articolelor și a datelor se realizează prin intermediul *rubricilor de descriere*, care reprezintă unitățile structurale ale diviziunii DATA. O rubrică de descriere, care sintactic este asemănătoare unei fraze din celelalte două diviziuni prezentate, grupează una sau mai multe clauze care precizează caracteristicile unui fișier sau ale unei date. Există deci două tipuri de rubrici: *rubrici de descriere a fișierelor* și *rubrici de descriere a datelor*. Primul tip poate apărea numai în secțiunea FILE, pe cînd al doilea tip, în toate secțiunile diviziunii DATA.

Formatul general al diviziunii DATA este următorul:

## DATA DIVISION

### [FILE SECTION

{rubrică de descriere a fișierelor.  
{rubrică de descriere a datelor.} ... } ...]

### [WORKING-STORAGE SECTION

{rubrică de descriere a datelor.} ...]

### [LINKAGE SECTION

{rubrică de descriere a datelor.} ...]

### [REPORT SECTION

{rubrică de descriere a datelor.} ...]

**Rubrici de descriere a fișierelor.** Formatul general, simplificat, al rubricilor de descriere a fișierelor (numite și rubrici FD) este următorul:

FD *nume-fișier* LABEL { RECORDS IS  
RECORDS ARE } OMITTED

[;RECORDING MODE IS { F  
V }].

Fiecărui fișier descris printr-o frază SELECT în paragraful FILE-CONTROL al diviziunii ENVIRONMENT, trebuie să-i corespundă, în secțiunea FILE, o rubrică FD. Clauzele rubricii FD indică:

- clauza LABEL: fișierului nu îi sînt asociate etichete de identificare;
- clauza RECORDING: fișierul are articole de format fix (opțiunea F) sau de format variabil (opțiunea V).

#### Observație

Fișierele pe cartele trebuie să aibă totdeauna articole de format fix, lungimea maximă fiind de 80 caractere. Fișierele la imprimantă pot avea articole de format fix sau variabil, lungimea maximă fiind de 133 de caractere. Primul caracter al articolelor fișierelor la imprimantă, numit *caracter de salt*, deși nu este tipărit trebuie să fie inclus în cadrul articolelor.

Clauza LABEL trebuie să figureze în oricare rubrică FD; în schimb, clauza RECORDING, poate fi omisă dacă formatul articolelor fișierului este variabil. În ceea ce privește sintaxa rubricii FD, trebuie făcută observația că indicatorul FD trebuie scris din coloana 8, iar numele fișierului (*nume-fișier*), din coloana 12.



**Rubrici de descriere a datelor.** Formatul general, simplificat, al rubricilor de descriere a datelor este următorul:

$$\text{număr-nivel} \left\{ \begin{array}{l} \text{nume-dată} \\ \text{FILLER} \end{array} \right\} \left[ ; \left\{ \begin{array}{l} \text{PICTURE} \\ \text{PIC} \end{array} \right\} \text{IS } \text{șablon} \right] \\ [; \text{VALUE IS } \text{literal}].$$

Prin intermediul rubricilor de descriere, datelor li se precizează următoarele caracteristici: numele, categoria, lungimea, valoarea și structura.

● **Numele** unei date reprezintă un cuvânt utilizator prin intermediul căruia data este identificată (referită) în cadrul programului.

Cuvântul rezervat **FILLER** poate fi asociat ca nume unei date care nu este utilizată în prelucrare; aceasta implică faptul că data respectivă nu poate fi referită (citată ca operand în instrucțiunile COBOL) în program.

● **Valoarea** unei date este reprezentată de caracterele care îi sînt asociate înaintea începerii execuției programului (valoare inițială), sau la un moment dat, în timpul execuției acestuia. În cazul în care valoarea unei date nu se schimbă în timpul prelucrării, data reprezintă o *constantă*.

În limbajul COBOL, constantele sînt de două tipuri: *literale* și *constante figurative*. Literalele pot reprezenta numere întregi sau reale, caz în care se numesc *numerice*, sau șiruri de caractere, caz în care se numesc *nenumeric*.

*Literalele numerice* pot fi exprimate:

— sub forma unor succesiuni de maximum 18 cifre (în care poate apărea și marca zecimală, însă nu după ultima cifră din dreapta), precedate, eventual, de semnul + sau —;

— sub forma  $[\pm] \text{ mantisă } E [\pm] \text{ exponent}$ , unde mantisa reprezintă o succesiune de maximum 16 cifre, care trebuie să conțină și marca zecimală, iar exponentul reprezintă un număr întreg format din una sau două cifre (valoarea algebrică a unui literal scris în această formă este dată de relația:  $\pm \text{ mantisa} \cdot 10^{\text{exponent}}$ ).

Exemple de literale numerice:

18;—5.26;.120;—15.2; 123.06E-2; —.12E10

*Literalele nenumeric* reprezintă succesiuni de maximum 120 caractere (cu excepția caracterului „apostrof”), delimitate prin apostrofuri.

Exemple de literale nenumeric:

'VALOAREA CALCULATA ESTE:.'; '—12.16'; 'DEMONTATI ROLA I02651'.

*Constantele figurative* reprezintă un tip de aparte constante, specifice limbajului COBOL, identificate prin nume simbolice, ce sînt cuvinte rezervate. Valorile desemnate prin constantele figurative sînt date de semnificația cuvintelor rezervate ce le-au fost asociate (tabelul XII.2)

#### Observații

Dacă o constantă figurativă este asociată unei date, de exemplu prin instrucțiunea de transfer **MOVE**, lungimea constantei figurative este egală cu lungimea datei. Instrucțiunea **MOVE SPACE TO ART-E**, din linia sursă 67 a programului XII.1, descrie operația de înregistrare, în fiecare locație a zonei asociate datei **ART-E**, a caracterului „spațiu”. Formele de singular și plural ale constantelor figurative sînt echivalente.



## Constante figurative

Constanta figurativă	Valoare
{ZERO ZEROS ZEROES}	unul sau mai multe caractere „zero”
{SPACE SPACES}	unul sau mai multe caractere „spațiu”
{HIGH-VALUE HIGH-VALUES}	unul sau mai multe caractere a căror reprezentare internă este $FF_{16}$
{LOW-VALUE LOW-VALUES}	unul sau mai multe caractere a căror reprezentare internă este $00_{16}$
{QUOTE QUOTES}	unul sau mai multe caractere „apostrof”
ALL 'c'	unul sau mai multe caractere desemnate prin <i>c</i>

● **Categoria** unei date este definită, în general, de valorile pe care data le poate lua, formatul de reprezentare internă a acestor valori, operațiile la care poate fi supusă data etc. În limbajul COBOL, există patru categorii de date:

— *date numerice*, ale căror valori reprezintă succesiuni de caractere numerice;

— *date alfabetice*, ale căror valori reprezintă succesiuni de caractere alfabetice;

— *date alfanumerice*, ale căror valori reprezintă succesiuni de caractere oarecare (numerice, alfabetice, speciale);

— *date numerice de editare*, ale căror valori reprezintă succesiuni de caractere numerice și speciale (adică numere pregătite pentru tipărire).

● **Lungimea** (dimensiunea) unei date reprezintă numărul caracterelor ce compun valorile datei.

● **Structura** datelor reprezintă modul de asociere și aranjare a datelor în funcție de semnificația lor.

În limbajul COBOL, din punctul de vedere al structurii, datele se împart în două tipuri: *date grupate* și *date elementare*. *Datele grupate* se compun din alte date, care la rândul lor pot fi alcătuite din alte date ș.a.m.d. În cadrul unei date grupate, datele care o compun pot fi dispuse pe mai multe nivele. Datele situate la un anumit nivel, care nu mai conțin alte date, se numesc *date elementare*.

În cadrul programelor COBOL, punerea în evidență a structurii datelor se realizează prin intermediul *numerelor de nivel*, care pot lua valori cuprinse între 01 și 49. Numărul de nivel 01 se asociază întotdeauna articolelor, iar celelalte numere de nivel, datelor care compun articolele. *Articolul* este, în general, o structură de date eterogenă, adică datele care îl compun nu sînt



neapărat de aceeași categorie. Un articol poate reprezenta o structură de date internă, existentă în memoria centrală, sau o structură de date externă, existentă pe un suport extern (cartelă, bandă magnetică etc.) și care reprezintă deci o înregistrare logică a unui fișier. În cadrul unui articol, datelor situate pe același nivel li se asociază același număr de nivel.

Schematic articolul ART-I ce aparține fișierului FIȘIER-INTRARE din programul XII.1, poate fi reprezentat ca în figura XII.2. Dacă însă se consideră că data ADRESA-I conține trei date — ORAS-I, STRADA-I și NR-I — atunci, structura articolului poate fi reprezentată ca în figura XII.3 iar descrierea acestuia s-ar putea face astfel:

```

01 ART-I.
05 CLASA-I   PIC X(6).
05 NUME-I    PIC A(20).
05 ADRESA-I.
10 ORAȘ-I    PIC X(15).
10 STRADA-I  PIC X(20).
10 NR-I      PIC 9(3).
05 MEDIE-I   PIC 99V99.

```

Analizînd descrierea articolului ART-I, prezentată anterior, și cea din programul XII.1, cît și a articolului SFIRSIT-FISIER, se pot face următoarele observații:

— în descrierea articolului ART-I din programul XII.1 există o singură dată grupată ART-I, celelalte fiind date elementare;

— în descrierea articolului ART-I, prezentată anterior, datele grupate sînt ART-I și ADRESA-I;

— rubricile de descriere asociate diferitelor date au fost scrise decalat spre dreapta, în funcție de numerele de nivel cu care încep; acest mod de scriere nu este impus de o regulă a limbajului, ci de practică, pentru a fi puse în evidență, și în acest mod, structura și ierarhia datelor;

— nu este obligatoriu ca numerele de nivel să crească numai cu o unitate de la un nivel al altul al structurii;

— este posibil ca un articol să nu conțină date subordonate.

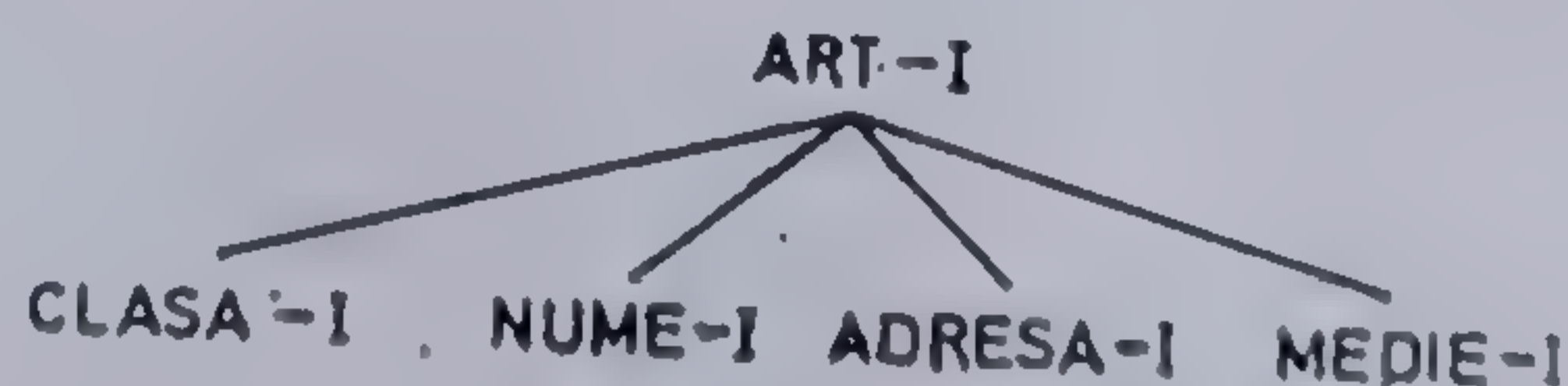


Fig. XII.2

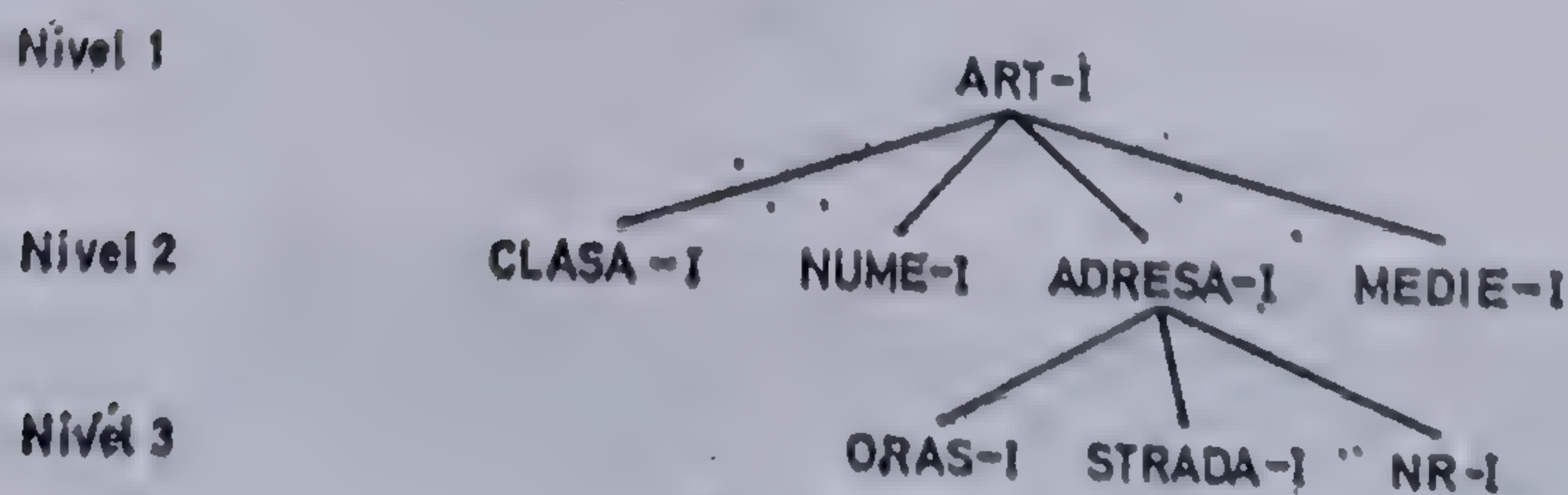


Fig. XII.3



## Observații

Datele elementare ce nu aparțin articolelor pot fi descrise în secțiunea WORKING-STORAGE în rubrici cu numere de nivel 77, ca în exemplul următor:

77 SUMA PIC 9(5).  
77 SF PIC 9.

Rubricile cu numere de nivel 77 trebuie să preceadă rubricile de descriere a articolelor.

În general, indentificatorii asociați datelor unui program sînt distincți. În cazul în care în program sînt descrise două sau mai multe articole ale căror date au aceiași identifikatori, referirea acestor date necesită *calificarea*. De exemplu, considerînd pentru articolele ART-I și ART-E următoarele descrieri:

01 ART-I.  
05 CLASA PIC X(6).  
05 NUME PIC A(20).  
05 ADRESA PIC X (38).  
05 MEDIE PIC 99V99.  
01 ART-E.  
05 CLASA PIC X(6).  
05 NUME PIC A(20).  
05 ADRESA PIC X(38).  
05 MEDIE PIC 99V99.

transferul valorii CLASA din articolul ART-I în zona asociată datei CLASA din articolul ART-E, poate fi descris astfel:

MOVE CLASA OF ART-I TO CLASA IN ART-E

Calificarea datelor este realizată cu ajutorul cuvintelor rezervate IN și OF, care asociază fiecărei date o dată grupată unic definită (căreia aparține data calificată), numită *calificator*.

Calificatorul de cel mai înalt nivel este fișierul; deci, în program pot fi descrise două sau mai multe fișiere care au aceiași identifikatori pentru articole (dacă un fișier are mai multe tipuri de articole, acestea trebuie însă să aibă identifikatori distincți); rezultă că numele fișierelor trebuie să fie unice. Articolele descrise în alte secțiuni decît secțiunea FILE, neputînd fi calificate, trebuie să li se asocieze identifikatori distincți; datele ce compun aceste articole pot avea aceiași identifikatori, calificarea făcîndu-se prin numele unor date în a căror componentă intră, sau prin numele articolelor.

● **Clauza PICTURE** precizează, prin intermediul șablonului, informații referitoare la categoria și lungimea unei date elementare. Șablonul reprezintă o succesiune de maximum 30 de simboluri, cu semnificații speciale; aceste simboluri definesc, în general, categoria datei, iar numărul lor, lungimea datei. Principalele simboluri care pot fi utilizate în șabloane sînt prezentate în tabelul XII.3.

În legătură cu utilizarea simbolurilor în cadrul șabloanelor pot fi făcute următoarele observații:

a) Marca zecimală virtuală nu este reprezentată în memorie, dar este luată în considerare de compilator la generarea codurilor ce corespund datei descrise; ea nu poate fi ultimul simbol al șablonului.

b) În cazul unei date numerice de editare, în locațiile zonei de memorie asociată datei, ce corespund pozițiilor simbolurilor  $\cdot$  și  $+$  din șablon, vor fi inserate, în momentul atribuirii de valori datei, caracterele  $\cdot$  și  $+$  sau  $-$ .



TABELUL XII.3

## Principalele simboluri permise în șabloanele de descriere

Categoria datei	Simbolul	Semnificația
numerică	9	indică poziția unei cifre
	V	indică poziția mărcii zecimale virtuale
	S	indică faptul că data reprezintă un număr cu semn
alfabetică	A	indică poziția unui caracter alfabetic
alfanumerică	X	indică poziția unui caracter alfanumeric
numerică de editare	9	indică poziția unei cifre
	.	indică poziția mărcii zecimale reale
	+	indică poziția în care va fi inserat caracterul +, dacă valoarea datei este un număr pozitiv, sau caracterul -, dacă valoarea datei este un număr negativ.

c) Simbolurile S și + pot figura numai o singură dată în șablon, și anume, la începutul acestuia.

d) Numărul maxim de simboluri 9 care pot apărea în șablon este 18.

e) În cazul în care un simbol se repetă succesiv într-un șablon, de un număr de ori, scrierea poate fi simplificată scriind simbolul o singură dată, iar imediat după el, numărul de repetări, între paranteze

De exemplu, descrierile următoare sînt echivalente:

05 CLASA-I PIC XXXXX.

05 CLASA-I PIC X(5).

● **Clauza VALUE** precizează, prin intermediul literalului, valoarea inițială a unei date elementare sau grupate, descrisă în secțiunea WORKING-STORAGE. Tipul literalului trebuie să coincidă cu tipul datei. Astfel, în cazul datelor elementare numerice, literalul trebuie să fie numeric sau constanta figurativă ZERO; în cazul datelor elementare nenumerice și a celor grupate, literalul trebuie să fie nenumeric sau una din constantele figurative SPACES, HIGH-VALUE, LOW-VALUE, QUOTE, ALL. De asemenea, lungimea literalului nu trebuie să depășească lungimea datei. Dacă lungimea literalului este mai mică decît lungimea datei, înregistrarea caracterelor ce compun valoarea literalului în zona-memorie asociată datei se realizează astfel:

— în cazul datelor elementare numerice, caracterele sînt înregistrate în zonă de la marca zecimală spre stînga și, eventual, spre dreapta, cu completare de zerouri;

— în cazul datelor elementare nenumerice sau a datelor grupate, caracterele sînt înregistrate în zonă de la stînga spre dreapta, cu completare de spații.

**Observație**

Dacă literalul citat în clauza VALUE este o constantă figurativă, în fiecare locație a zonei ste înregistrat caracterul desemnat de constanta figurativă.

În tabelul XII.4. sînt prezentate cîteva descrieri de inițializări de date și imaginea zonelor asociate acestora.



TABELUL XII.4

## Exemple de inițializare de date

Descrierea datei	Imaginea zonei, după inițializare
02 DATA-1 PIC X(4) VALUE 'A.1C'.	A   .   1   C
02 DATA-2 PIC A(4) VALUE ALL 'X'.	X   X   X   X
02 DATA-3 PIC 9(4) VALUE 12.	0   0   1   2
02 DATA-4 PIC 99V99 VALUE 1.2.	0   1   2   0
02 DATA-5 PIC S99V99 VALUE 0.55.	0   0   5   5
02 DATA-6 PIC X(4) VALUE 'AB'.	A   B

În ceea ce privește sintaxa rubricilor de descriere a datelor se pot face următoarele observații:

— numărul de nivel trebuie să fie urmat de numele datei, iar acesta, de clauzele care precizează caracteristicile datei (ordinea în care sînt citate clauzele poate fi oarecare);

— numerele de nivel 01 trebuie scrise din coloana 8 a formularului, iar numele articolelor, din coloana 12;

— numerele de nivel cu valori cuprinse între 02 și 49 pot fi scrise începînd din coloana 8, iar numele datelor trebuie să urmeze după cel puțin un spațiu (în intervalul cuprins între coloanele 12 și 72).

d. **PROCEDURE DIVISION** (diviziunea de prelucrare). Grupează instrucțiunile care descriu algoritmul prelucrării. *Unitățile structurale ale acestei diviziuni sînt secțiunile, paragrafele și frazele.*

Spre deosebire de secțiunile celorlalte diviziuni, care sînt identificate prin cuvinte rezervate și grupează informații prestabilite, numele și conținutul *secțiunilor* diviziunii **PROCEDURE** sînt stabilite de utilizator. Utilizarea secțiunilor în cadrul acestei diviziuni este, în general, opțională, fiind impusă numai de anumite situații: segmentarea programelor, sortarea fișierelor etc.

*Paragrafele* sînt identificate prin cuvinte utilizator, numite *nume de paragraf*, și conțin una sau mai multe fraze.

O *frază* poate conține una sau mai multe instrucțiuni COBOL (unite prin separatorii *punct-virgulă*, *virgulă*, *spațiu* sau *cuvîntul rezervat THEN*) și se termină printr-un punct urmat de spațiu.

O *instrucțiune* COBOL reprezintă un ansamblu de cuvinte (cuvinte rezervate și cuvinte utilizator) și, uneori, simboluri. Primul cuvînt este un cuvînt rezervat, numit *verb COBOL*, care indică operația (operațiile) ce va fi efectuată prin execuția instrucțiunii. Celelalte cuvinte reprezintă *operandi* și *cuvinte de legătură*. Operandii sînt cuvinte utilizator care desemnează, în general, datele implicate în operația descrisă de instrucțiune. Cuvintele de legătură sînt cuvinte rezervate (cheie sau opționale) al căror rol este de a defini acțiuni suplimentare impuse de operația indicată de instrucțiune. În cazul anumitor instrucțiuni, operandii și cuvintele de legătură pot forma clauze. De asemenea, anumite instrucțiuni pot include alte instrucțiuni.

De exemplu, instrucțiunea **READ** din linia sursă 55 a programului XII.1 descrie operația de citire a unui articol din fișierul FIȘIER-INTRARE. Verbul COBOL este cuvîntul rezervat **READ**. Cuvîntul utilizator FIȘIER-



INTRARE indică operandul instrucțiunii, iar clauza AT END precizează faptul că, dacă prin citire este detectat sfârșitul fișierului, datei SFÎRȘIT-FIȘIER i se atribuie (prin instrucțiunea MOVE) valoarea DA.

Formatul general, simplificat, al diviziunii PROCEDURE este următorul:

```
PROCEDURE DIVISION.  
{[nume-sectiune SECTION.]  
 {nume-paragraf.  
   {frază.} ... } ... }
```

Regulile de sintaxă referitoare la unitățile structurale ale diviziunii PROCEDURE sînt următoarele:

- numele de secțiuni și numele de paragrafe, numite *nume de prelucrare*, sînt cuvinte utilizator care se formează după regulile cunoscute; spre deosebire de celelalte cuvinte utilizator, ele pot fi formate și numai din cifre;
- diviziunea PROCEDURE trebuie să înceapă obligatoriu printr-un *nume de secțiune* sau de paragraf;
- numele secțiunilor și paragrafelor se scriu pe formular din coloana 8, iar frazele în continuarea numelor paragrafelor, sau pe linia următoare însă cel puțin din coloana 12.

În diviziunea PROCEDURE a programului XII.1 apar nouă instrucțiuni: OPEN, CLOSE, READ, WRITE, DISPLAY, IF, PERFORM, MOVE și STOP. Formatele generale ale acestor instrucțiuni și operațiile pe care le descriu sînt prezentate în continuare.

*Instrucțiunea OPEN.* Descrie operația de deschidere a unui sau a mai multor fișiere.

Format general:

```
OPEN { { INPUT  
        OUTPUT } nume-fișier ... } ...
```

Opțiunile INPUT și OUTPUT indică tipul prelucrării fișierului: consultare (citire) și, respectiv, creare (scriere). Citirea sau scrierea unui articol din/în fișier impune ca acesta să fi fost, în prealabil, deschis.

*Instrucțiunea CLOSE.* Descrie operația de închidere a unui sau a mai multor fișiere.

Format general.

```
CLOSE {nume-fișier} ...
```

Înainte de sfârșitul execuției programului, orice fișier care a fost deschis trebuie închis.

*Instrucțiunea READ.* Descrie operația de citire a unui articol dintr-un fișier secvențial.

Format general:

```
READ nume-fișier AT END instrucțiune.
```

După citire, articolul este disponibil pentru prelucrare, în zona-articol asociată fișierului. La fiecare execuție a acestei instrucțiuni se verifică dacă înregistrarea citită reprezintă marca de sfârșit de fișier (în cazul fișierelor pe cartele aceasta reprezintă o cartelă care conține, în primele patru coloane, caracterele .EOF). În caz afirmativ este executată instrucțiunea (sau secvența de instrucțiuni) precizată prin clauza AT END; în caz contrar, execuția programului continuă cu instrucțiunea ce urmează lui READ (de după punct).



### Observație

Pentru obținerea unui program corect structurat, fiecărui fișier secvențial consultat  $i$  se poate asocia un indicator căruia, înainte de efectuarea primei citiri  $i$  se atribuie o valoare  $v_1$ , iar în momentul în care este detectată marca de sfârșit de fișier, valoarea  $v_2$ ; citirea trebuie urmată de testarea valorii indicatorului, lucru care permite separarea (punerea în evidență) secvențelor ce corespund detectării sau nu a mărcii de sfârșit de fișier (fig. XII.4).

În cazul programului XII.1, indicatorul asociat fișierului FIȘIER-INTRARE este SFÎRȘIT-FIȘIER, căruia  $i$  s-a atribuit, prin clauza VALUE, valoarea inițială NU. La detectarea mărcii de sfârșit de fișier, acestui indicator  $i$  s-a atribuit, prin instrucțiunea MOVE, valoarea DA.

**Instrucțiunea WRITE.** Descrie operația de scriere a unui articol într-un fișier. În cazul fișierelor la imprimantă, formatul general al acestei instrucțiuni este:

**WRITE** *nume-articol* **AFTER ADVANCING** *întreg* **LINES**

Clauza ADVANCING precizează condițiile în care va avea loc scrierea articolului (rîndului) și anume după avansarea hîrtiei cu un număr de rînduri (interlinii) egal cu valoarea lui *întreg*; *întreg* poate lua valori cuprinse între 0 și 99, a căror semnificație este:

0 — hîrtia avansează pînă la începutul unei noi pagini;

1 — hîrtia avansează cu un rînd;

2 — hîrtia avansează cu două rînduri ș.a.m.d.

Scrierea rîndului trebuie să fie precedată de formarea imaginii acestuia, în zona-articol asociată fișierului.

**Instrucțiunea DISPLAY.** Descrie, în general, operația de tipărire la imprimantă a unui rînd ce conține valorile uneia sau mai multor date și/sau literale; lungimea maximă a rîndului este de 120 caractere.

Format general:

**DISPLAY**  $\left\{ \begin{array}{l} \text{nume-dată-1} \\ \text{literal-1} \end{array} \right\} \left[ \left\{ \begin{array}{l} \text{nume-dată-2} \\ \text{literal-2} \end{array} \right\} \right] \dots$

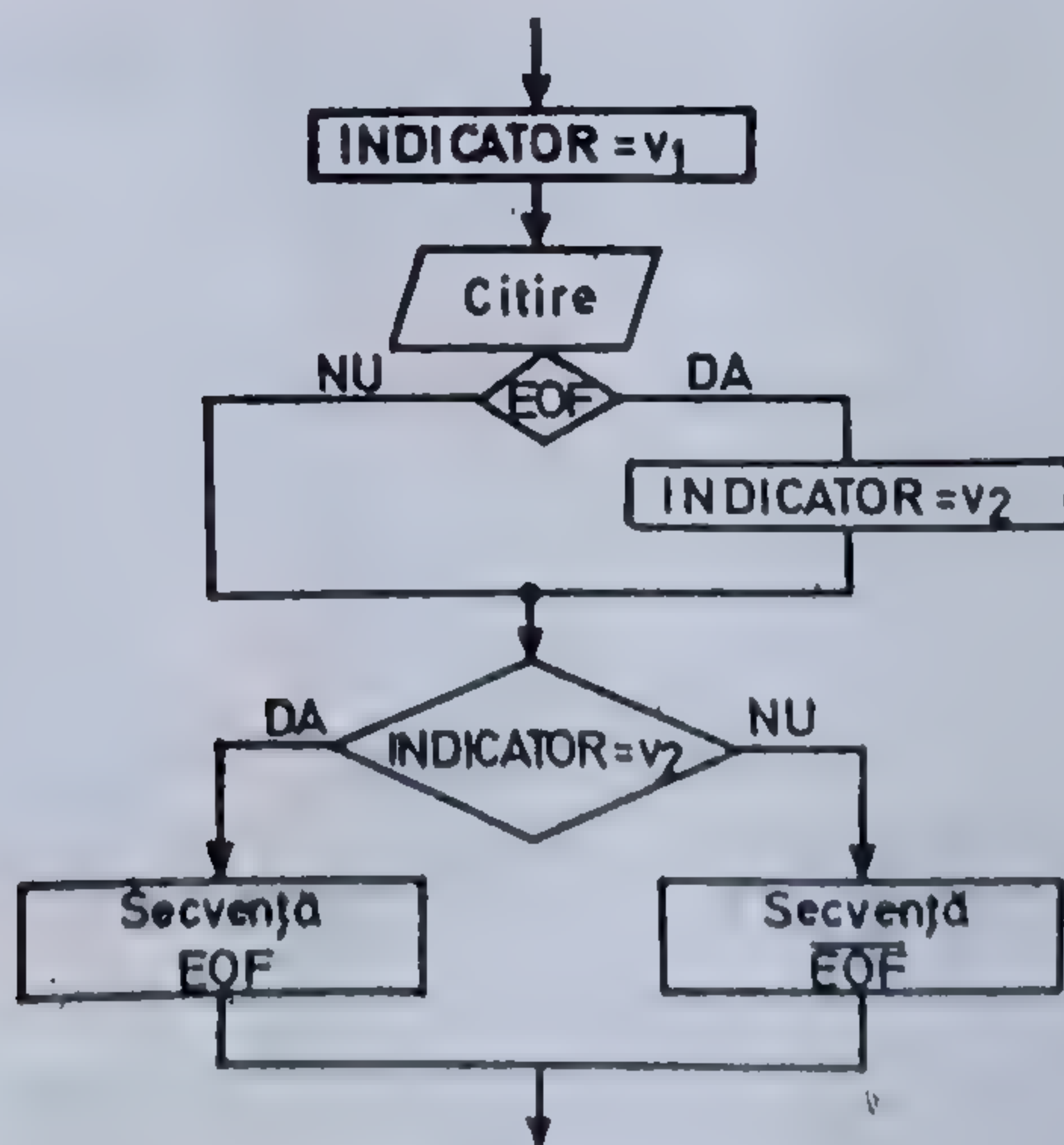


Fig. XII.4



*Exemplu.* Considerînd că valorile a două date DATA-1 și DATA-2 sînt 120 și, respectiv, 150, prin execuția instrucțiunilor:

```
DISPLAY 'VALOAREA DATEI ESTE = 'DATA-1
DISPLAY DATA-1 DATA-2
DISPLAY DATA-1' 'DATA-2
```

la imprimantă sînt tipărite trei rînduri, cu următorul conținut:

```
VALOAREA DATEI ESTE = 120
120150
120 150
```

*Instrucțiunea MOVE.* Descrie operația de transfer a valorii unei date, numită *dată emițătoare*, în zona asociată altei date, numită *dată receptoare*.

Format general:

**MOVE** { *nume-dată-1* } **TO** *nume-dată-2*  
*literal*

Considerînd că data emițătoare (*nume-dată-1* sau *literal*) și data receptoare (*nume-dată-2*) sînt de același tip înregistrarea caracterelor ce compun valoarea datei emițătoare în zona asociată datei receptoare se realizează astfel:

— în cazul datelor numerice, caracterele sînt înregistrate în zonă de la marca zecimală spre stînga și, eventual spre dreapta, cu completare de zerouri sau trunchiere de caractere;

— în cazul datelor nenumерice sau grupate, caracterele sînt înregistrate în zonă de la stînga spre dreapta, cu completare de spații sau trunchiere de caractere. În tabelul XII.5 sînt prezentate cîteva operații de transfer.

TABELUL XII.5

Data emițătoare		Data receptoare	
Șablon	Valoare	Șablon	Valoare
99V99	1   2   3   4	999V999	0   1   2   3   4   0
999V9	1   2   3   4	99	2   3
XXX	A   B   C	99999	0   0   A   B   C
XXX	A   B   C	XXXXXX	A   B   C
XXXX	A   B   C   D	XX	A   B
999	1   2   3	XXXXXX	1   2   3

*Instrucțiunea IF.* Realizează testarea unei condiții, numită în limbajul COBOL și test.



Format general:

```
IF condiție  
  THEN  
    instrucțiune-1  
  ELSE  
    instrucțiune-2.
```

În general, condiția reprezintă un test de relație, prin care se verifică relația — „mai mic“, „egal“ sau „mai mare“ — dintre doi termeni. Termenii pot fi ambii date de același tip, sau unul dintre termeni o dată, iar celălalt, un literal (ce corespunde ca tip datei). Operatorul de relație, indicat între cei doi termeni, poate fi simbolul  $<$ ,  $=$  sau  $>$ ; pe formular, operatorul de relație se scrie precedat și urmat de cel puțin un spațiu.

Considerînd secvențele din figura XII.5, ele pot fi codificate în limbajul COBOL astfel:

```
IF A = B  
  THEN  
    MOVE A TO C  
  ELSE  
    MOVE B TO C.  
IF A > B  
  THEN  
    MOVE A TO C  
    MOVE Y TO X.
```

Analizînd aceste exemple, se poate deduce principiul de execuție al instrucțiunii IF:

— dacă valoarea logică a condiției testate este „adevărat“, se execută instrucțiunea sau secvența de instrucțiuni indicată prin *instrucțiune-1*;

— dacă valoarea logică a condiției testate este „fals“, se execută instrucțiunea sau secvența de instrucțiuni indicată prin *instrucțiune-2*.

#### Observație

Valoarea logică a condiției testate poate fi negată prin operatorul logic NOT. De exemplu, prima instrucțiune IF, prezentată anterior, s-ar putea scrie și astfel:

```
IF A NOT = B  
  THEN  
    MOVE B TO C  
  ELSE  
    MOVE A TO C.
```

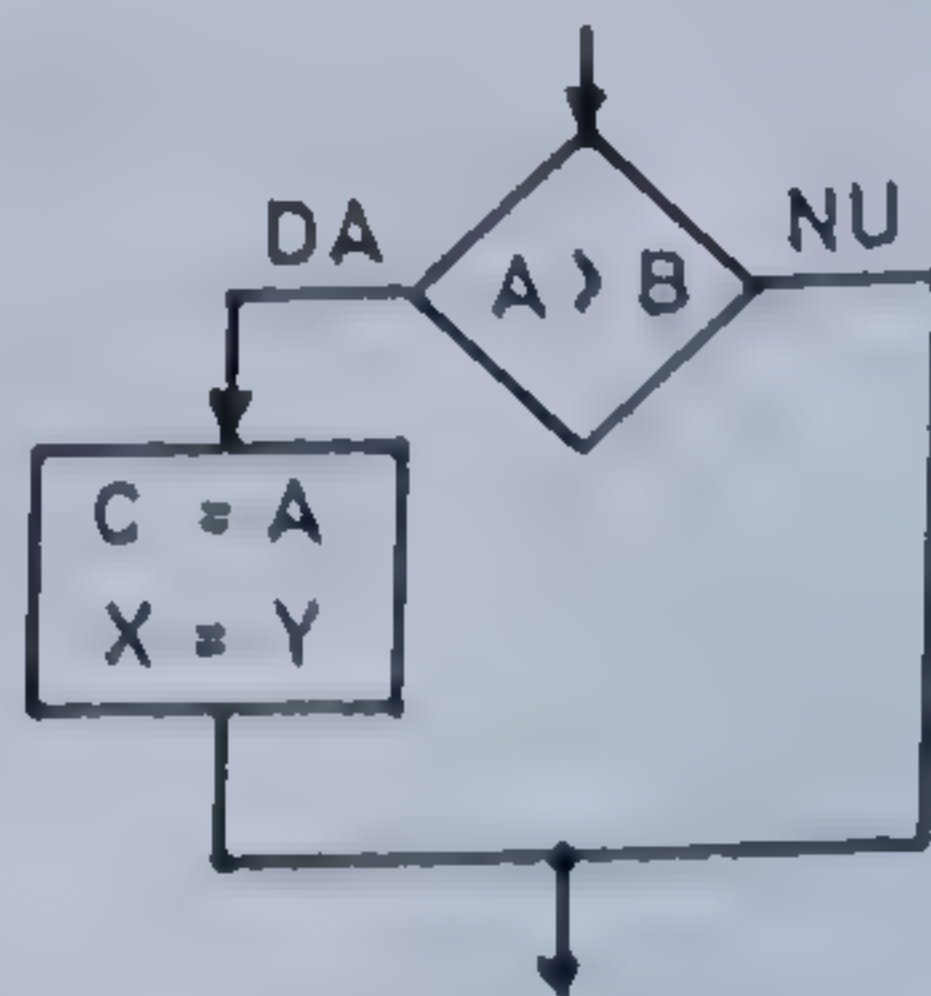
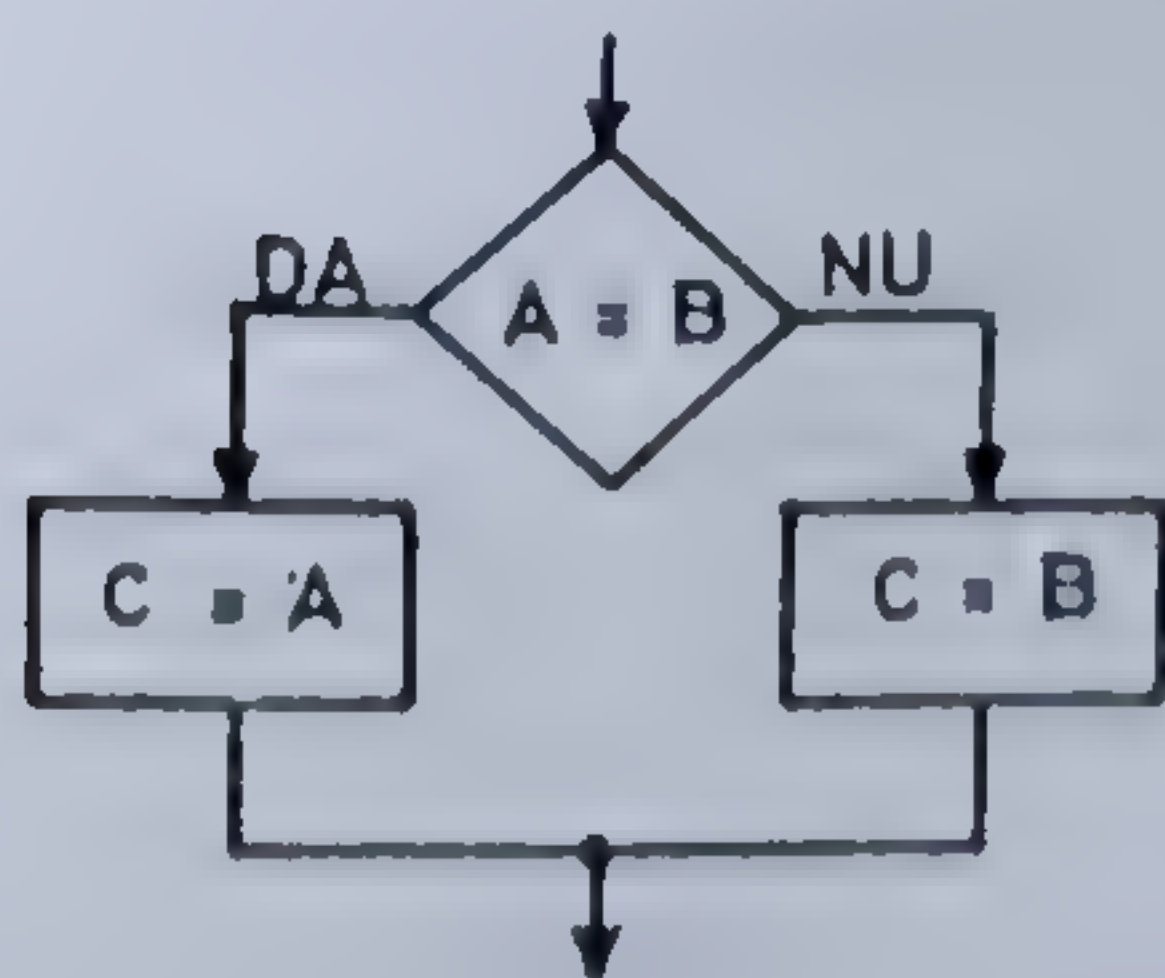


Fig. XII. 5



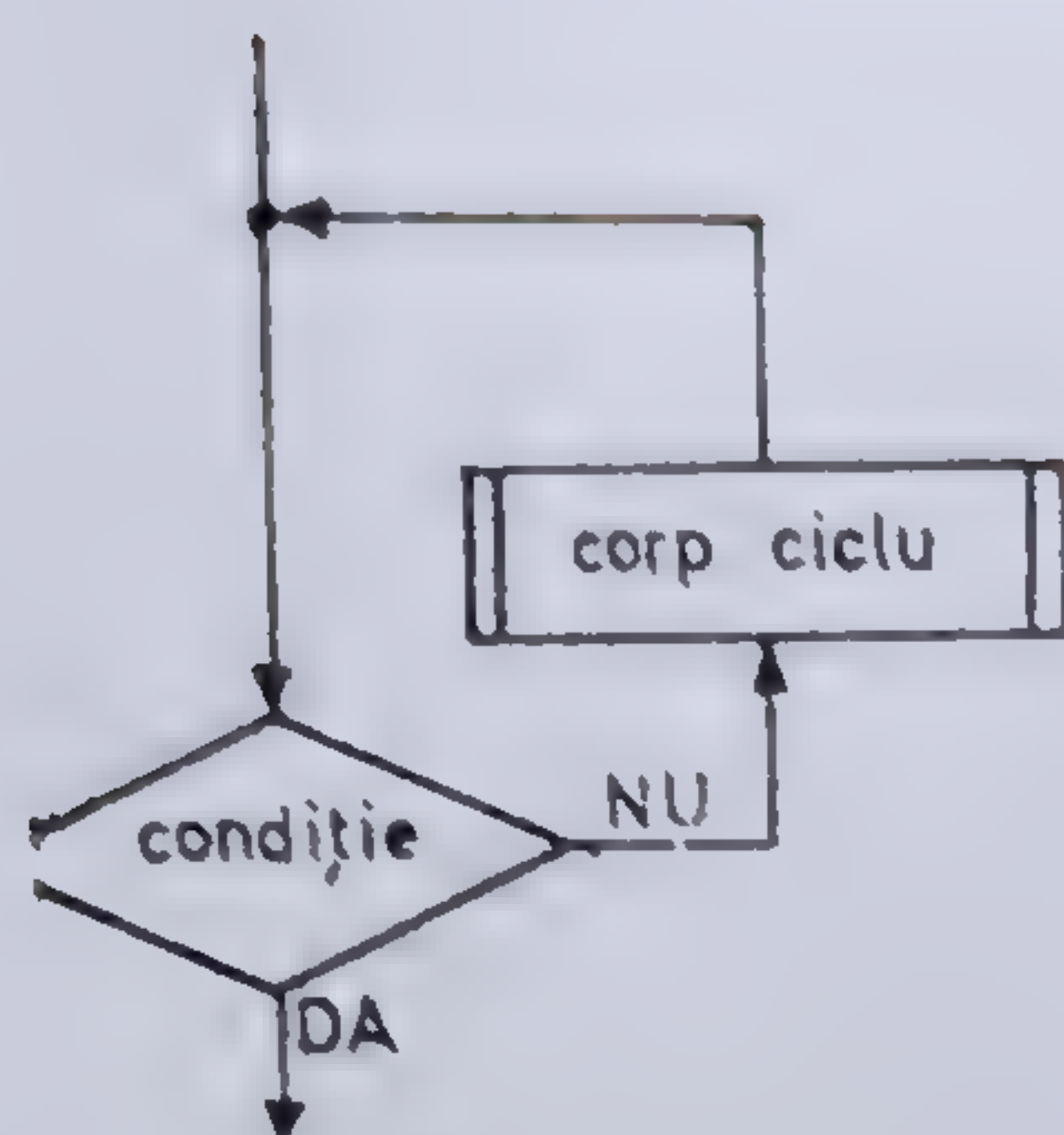


Fig. XII.6

**Instrucțiunea PERFORM UNTIL.** Descrie un ciclu de tipul celui prezentat în figura XII.6:

Format general:

**PERFORM** *nume-prelucrare* **UNTIL** *condiție*  
unde:

— *nume-prelucrare* indică corpul ciclului, care, în general, este un paragraf ce grupează una sau mai multe instrucțiuni;

— *condiție* reprezintă, în general, un test de relație de tipul celui prezentat la instrucțiunea IF.

Instrucțiunea PERFORM descrie, în ordine, următoarele operații:

— evaluarea condiției;

— execuția corpului ciclului atâta timp cât valoarea logică a condiției testate este „fals”;

— ieșirea din ciclu (saltul la instrucțiunea ce urmează lui PERFORM), dacă valoarea logică a condiției testate este „adevărat”.

**Instrucțiunea STOP.** Descrie operația de oprire definitivă sau temporară a execuției programului.

Format general:

**STOP** { *RUN* }  
          { *literal* }

Oprirea definitivă a execuției programului este indicată prin opțiunea RUN, iar oprirea temporară a execuției, prin opțiunea *literal*. În cazul primei opțiuni, prin execuția instrucțiunii STOP controlul este transferat sistemului de operare care va trata următoarea fază a lucrării. În cazul celei de a doua opțiuni, prin execuția instrucțiunii STOP caracterele care compun literalul sînt tipărite la mașina de scris și execuția programului este oprită; reluarea execuției poate fi determinată de intervenția operatorului la pupitrul de comandă. În general, oprirea temporară a execuției se face cu scopul de a permite operatorului să efectueze anumite operații (montări sau demontări de volume etc.); valoarea literalului, a cărui lungime nu trebuie să depășească 72 de caractere, comunică operatorului operațiile pe care trebuie să le efectueze, ca de exemplu:

**STOP 'MONTAȚI ROLA XY0102 PE UNITATEA MT1'**

Sistematizînd noțiunile prezentate în cadrul acestui paragraf se pot face următoarele precizări:

● Limbajul COBOL poate fi definit ca o mulțime (infinită) ale cărei elemente sînt programe COBOL.

● Programele COBOL reprezintă succesiuni de elemente ale vocabularului COBOL, obținute conform gramaticii limbajului.

● Elementele care compun vocabularul limbajului — numite cuvînte COBOL — sînt de șase tipuri:

- caractere de bază;
- cuvinte rezervate;
- cuvinte utilizator;
- literale;
- numere de nivel;
- șabloane.



● Un program COBOL este împărțit, din punct de vedere sintactic, în diviziuni, secțiuni, paragrafe, rubrici (fraze), clauze, instrucțiuni (fig. XII.7).

● Scrierea programelor COBOL se realizează pe un formular, cu o liniatură specială, care conține 80 de coloane și un număr oarecare de linii (fig. XII.8). Cele 80 de coloane sînt grupate în patru zone distincte, după cum urmează:

— *zona de secvență* (coloanele 1–6), utilizată pentru numerotarea liniilor programului; completarea acestei zone este opțională;

— *zona de continuare* (coloana 7), care poate conține caracterul „liniuță” (care marchează continuarea liniei precedente), caracterul „asterisc” (care indică o linie comentariu) sau caracterul „spațiu” (care indică o linie sursă obișnuită);

— *zona enunțului COBOL* (coloanele 8–72), destinată scrierii textului sursă propriu-zis; această zonă se divide în alte două zone numite zona A (coloanele 8–11) și respectiv, zona B (coloanele 12–72); coloanele 8 și 12 se numesc marginea A și, respectiv, marginea B;

— *zona de identificare* (coloanele 73–80), care poate fi utilizată pentru identificarea programului.

● Scrierea corectă a programelor COBOL pe formular impune respectarea următoarelor reguli:

— Din marginea A se scriu titlurile diviziunilor și secțiunilor, numele paragrafelor, indicatorul FD al rubricii de descriere a fișierelor, numărul de nivel 01 din rubricile de descriere a articolelor;

— Începînd din marginea B se scriu frazele ce compun paragrafele (în continuarea numelor paragrafelor sau pe linia următoare), numele fișierelor citate în rubrica FD, numele articolelor citate în rubricile cu număr de nivel 01;

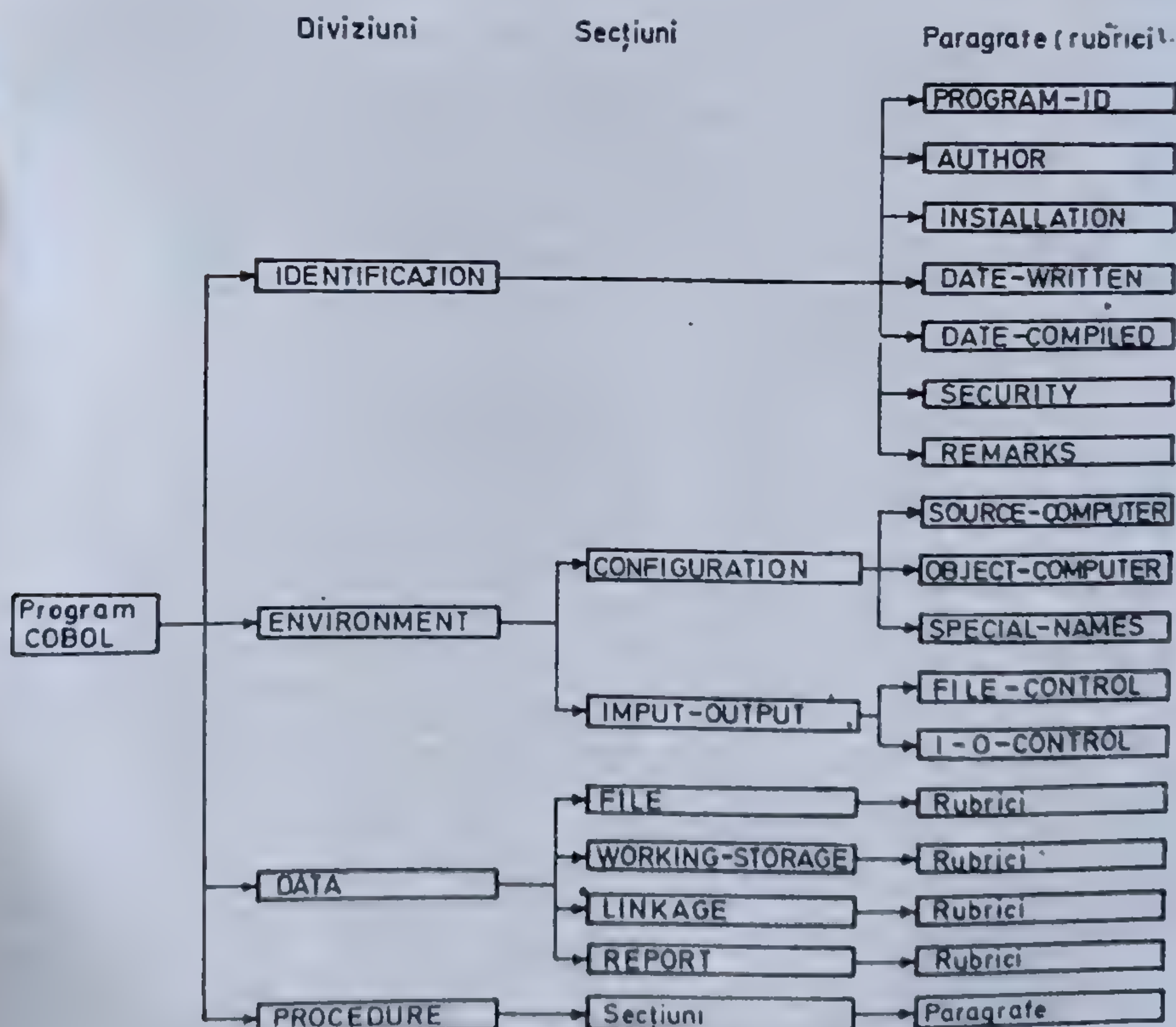


Fig. XII.7







operandilor citați într-o instrucțiune aritmetică este mai mare decât doi, compilatorul rezervă automat zone-memorie pentru memorarea valorilor rezultatelor intermediare.

a. **Clauzele rounded și size error.** În formatele generale ale instrucțiunilor aritmetice există două clauze optionale: **ROUNDED** și **SIZE ERROR**.

● **Clauza ROUNDED** indică rotunjirea valorii rezultatului obținut în urma efectuării unei operații aritmetice. În absența clauzei **ROUNDED**, dacă numărul de zecimale ale valorii rezultatului este mai mare decât numărul de locații rezervat părții zecimale a acestuia, are loc o trunchiere (fig. XII.10 a). În prezența clauzei **ROUNDED**, ultima zecimală a valorii rezultatului (zecimală care poate fi înregistrată în zona asociată acestuia) este majorată cu o unitate, dacă zecimala care îi urmează este mai mare sau egală cu 5 (fig. XII.10 b).

● **Clauza SIZE ERROR** indică testarea depășirilor de capacitate. Format general:

**ON SIZE ERROR** *instrucțiune*.

O depășire de capacitate apare în cazul în care lungimea zonei-memorie în care trebuie memorată valoarea rezultatului final al unei operații aritmetice nu permite înregistrarea acestei valori (de asemenea, este asimilată unei depășiri de capacitate și o împărțire la zero). Depășirile de capacitate pot fi semnalate, în cursul execuției programelor COBOL, numai când clauza **SIZE ERROR** figurează în instrucțiunile aritmetice; dacă această clauză lipsește și apar depășiri, valorile rezultatelor finale sînt înregistrate trunchiat, în zonele care le sînt asociate. În figura XII.11 este prezentat principiul de execuție a unei instrucțiuni aritmetice, iar figura XII.12, patru operații de adunare. În cazul operațiilor din figurile XII.12 a și XII.12 b. — rezultatele obținute sînt corecte deoarece, prin memorarea valorilor acestora, se pierde o cifră nesemnificativă — zero — și respectiv, ultima cifră a părții zecimale;

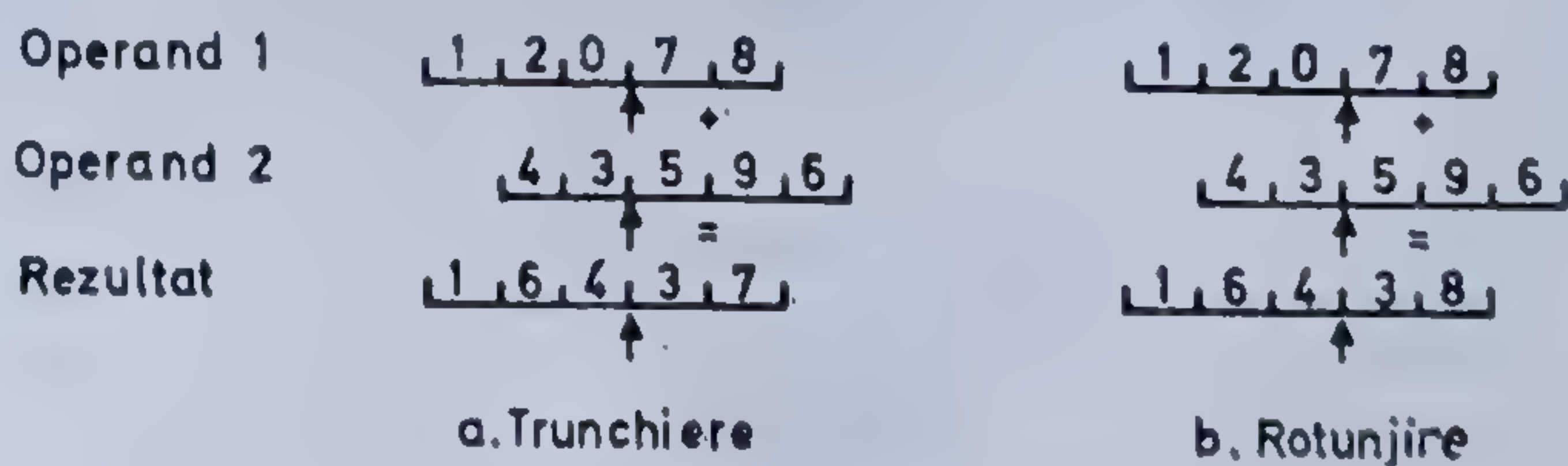


Fig. XII.10

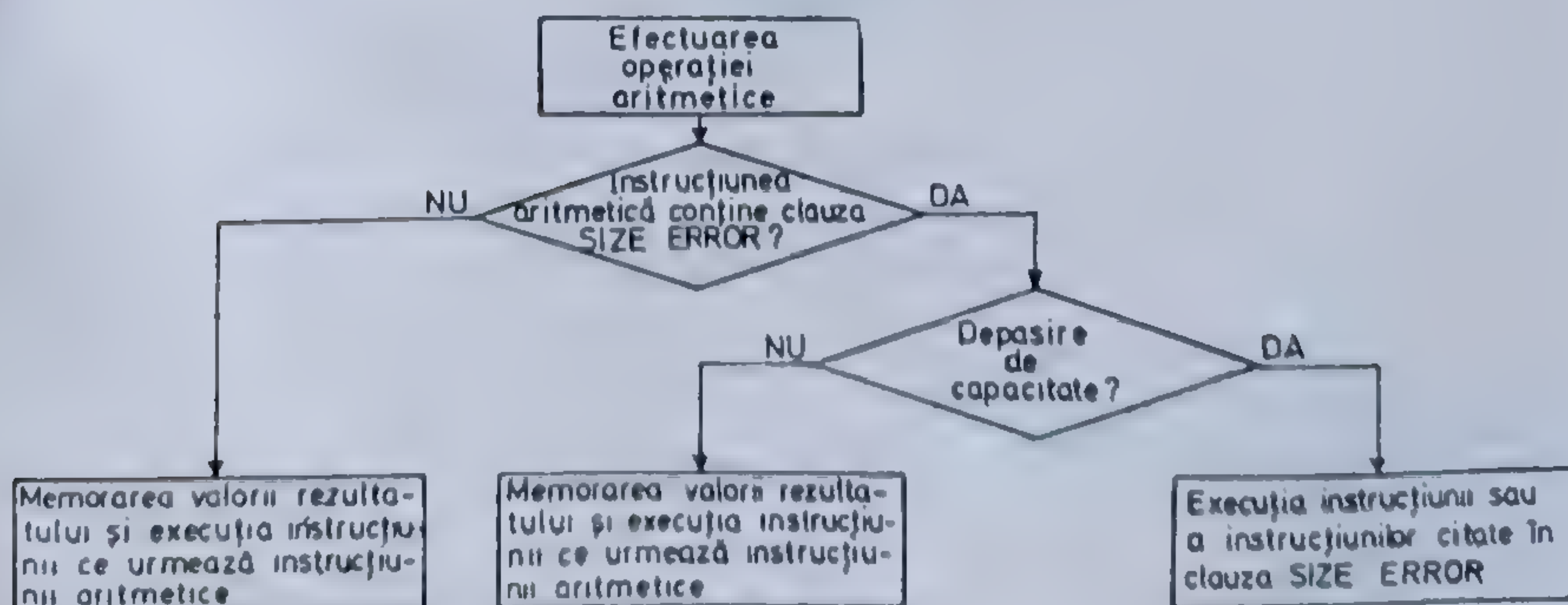


Fig. XII.11



Operand 1	$\begin{array}{ c c c } \hline 0 & 1 & 2 \\ \hline \end{array}$	$\begin{array}{ c c c c } \hline 0 & 1 & 7 & 8 \\ \hline \end{array}$	$\begin{array}{ c c c } \hline 0 & 1 & 2 \\ \hline \end{array}$	$\begin{array}{ c c c } \hline 0 & 1 & 2 \\ \hline \end{array}$
Operand 2	$\begin{array}{ c c c } \hline 0 & 3 & 4 \\ \hline \end{array}$	$\begin{array}{ c c c } \hline 5 & 6 & 2 \\ \hline \end{array}$	$\begin{array}{ c c c } \hline 0 & 9 & 4 \\ \hline \end{array}$	$\begin{array}{ c c c } \hline 0 & 9 & 4 \\ \hline \end{array}$
Rezultat	$\begin{array}{ c c } \hline 4 & 6 \\ \hline \end{array}$	$\begin{array}{ c c c } \hline 5 & 7 & 9 \\ \hline \end{array}$	$\begin{array}{ c c } \hline & \\ \hline \end{array}$	$\begin{array}{ c c } \hline 0 & 6 \\ \hline \end{array}$
	a. Rezultat corect	b. Rezultat corect	c. Depășire	d. Trunchiere

Fig. XII.12

În cazul operațiilor din figurile XII.12 c și XII.12 d este semnalată o depășire, sau valoarea rezultatului este trunchiată, după cum clauza SIZE ERROR figurează sau nu în instrucțiunea aritmetică.

#### Observație

Inserarea în program a unei secvențe (proceduri) pentru tratarea depășirilor necesită utilizarea unui indicator, căruia trebuie să i se atribuie o valoare  $v_1$ , înainte de efectuarea operației aritmetice, și o valoare  $v_2$ , dacă este semnalată o depășire. Prin testarea valorii acestui indicator, care se realizează printr-o instrucțiune IF ce urmează instrucțiunii aritmetice, pot fi separate secvențele de instrucțiuni ce vor fi executate dacă s-a produs sau nu depășire (fig. XII.3).

b. **Expresii aritmetice.** În limbajul COBOL expresiile aritmetice se formează din operanzi (termeni), operatori și, eventual, paranteze. *Operanzii* reprezintă date elementare numerice sau literale numerice.

*Operatorii* care indică operațiile ce trebuie efectuate asupra operanzilor, sînt următorii:

- + pentru operația de adunare;
- pentru operația de scădere;
- \* pentru operația de înmulțire;
- / pentru operația de împărțire;
- \*\* pentru operația de ridicare la putere.

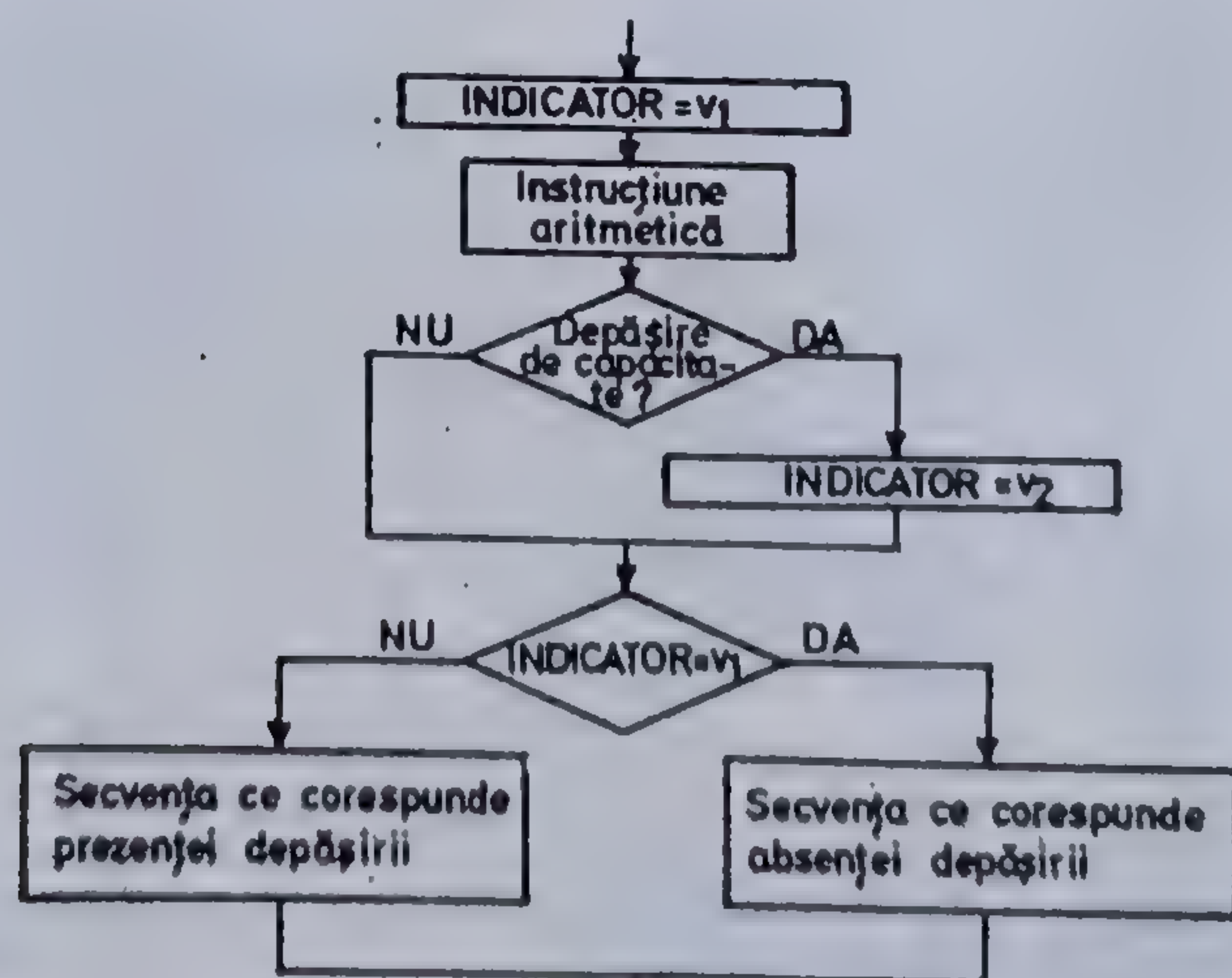


Fig. XII.13



În absența parantezelor, ordinea executării operațiilor unei expresii aritmetice este următoarea:

- 1 — ridicarea la putere  
2 — înmulțirea și împărțirea  
3 — adunarea și scăderea.

Prezența parantezelor poate modifica această ordine; în acest caz evaluarea expresiei începe cu parantezele interioare.

Scierea expresiilor aritmetice impune respectarea următoarelor reguli:

— o expresie aritmetică nu poate începe decât cu simbolurile  $($ ,  $-$  sau cu un operand și nu se poate termina decât cu simbolul  $)$  sau un operand;

— operatorii trebuie să fie precedați și urmați de cel puțin un spațiu;

— o paranteză deschisă nu poate fi urmată de spațiu, dar trebuie precedată de cel puțin un spațiu;

— o paranteză închisă nu poate fi precedată de spațiu, dar trebuie urmată de cel puțin un spațiu.

În figura XII.14 este prezentat modul de scriere a expresiilor:

$$A \cdot 3,14 - \sqrt{B}$$

SUMA + CANTITATE \* PREȚ

$$\sqrt{A^2} - \sqrt{B^2}$$

c. Instrucțiuni aritmetice. *Instrucțiunea COMPUTE* descrie operația prin care unei date (*nume-dată-1*) i se atribuie valoarea altei date (*nume-dată-2* sau *literal*) sau valoarea unei expresii aritmetice (*expresie-aritmetică*).

### Format general:

**COMPUTE** *nume-dată-1*

$$[\text{ROUNDED}] = \left\{ \begin{array}{l} \text{nume-dată-2} \\ \text{literal} \\ \text{expresie-aritmetică} \end{array} \right\}$$

[ON SIZE ERROR *instructione.*]

Data *nume-dată-1* poate fi numerică sau numerică de editare. Numai în primul caz data *nume-dată-1* poate apărea și ca operand într-o expresie aritmetică. De exemplu, considerînd descrierea unui articol:

## 01 OPERANZI REZULTATE.

05 A PIC S9(5).

05 B PIC S9(6).

05 R1 PIC S9(7).

05 R2 PIC +9(7).

[illegible]

**Fig. XII.14**



instrucțiunile:

```
COMPUTE R1 = A + B  
COMPUTE R2 = A + B
```

sînt corecte, pe cînd instrucțiunea

```
COMPUTE R2 = R2 + A
```

este incorectă.

### *Exemplu*

Modul de evaluare a expresiilor aritmetice în limbajul COBOL este ilustrat de programul XII.2, care determină valoarea polinomului:

$$P(X) = A \cdot X^3 + B \cdot X^2 + C \cdot X + D$$

pentru o anumită valoare a lui X și imprimă această valoare.

Valoarea lui X și valorile coeficienților A, B, C și D sînt reprezentate pe cartele astfel:

- în coloanele 1—2, valoarea lui X;
- în coloanele 3—6, valoarea coeficientului A, formată dintr-o parte întreagă (2 cifre) și o parte zecimală (2 cifre);
- în coloanele 7—8, valoarea coeficientului B;
- în coloanele 7—13, valoarea coeficientului C, formată dintr-o parte întreagă (2 cifre) și o parte zecimală (3 cifre);
- în coloanele 14—15, valoarea coeficientului D.

Valorile acestor date reprezintă numere cu semn; în plus, valorile datelor A și C au și parte zecimală. În legătură cu modul de reprezentare a valorilor datelor pe cartele se fac următoarele observații:

- valorile datelor numerice trebuie aliniate la dreapta în cadrul cîmpurilor ce le sînt rezervate, punîndu-li-se în evidență și zerourile nesemnificative:

```
ID DIVISION.  
PROGRAM-ID.  
    CALCUL..  
ENVIRONMENT DIVISION.  
CONFIGURATION SECTION.  
DATA DIVISION.  
WORKING-STORAGE SECTION.  
01  DATE-INTRARE.  
    05  X          PIC S99. ..  
    05  A          PIC S99V99.  
    05  B          PIC S99. ..  
    05  C          PIC S99V999.  
    05  D          PIC S99.  
01  VALOARE       PIC +9(9).9(5).  
PROCEDURE DIVISION  
CITESTE.  
    ACCEPT DATE-INTRARE.  
CALCULEAZĂ.  
    COMPUTE VALOARE = A * X ** 3 + B * X ** 2 + C * X + D  
SCRIE.  
    DISPLAY 'VALOARE POLINOM = ' VALOARE  
STOP RUN.
```

### PROGRAMUL XII.2



— semnul este indicat, numai în cazul valorilor negative, printr-o perforație în linia 11 a coloanei în care este perforată ultima cifră din dreapta (deci în această coloană vor exista două perforații); pe formularul de programare, semnul valorilor negative se indică prin caracterul — (liniuță), scris deasupra ultimei cifre din dreapta (fig. XII.15).

— marca zecimală nu se reprezintă pe cartelă, ea indicându-se în șablonul de descriere a datelor, prin simbolul V;

Datele de intrare au fost grupate în articolul DATE-INTRARE, descrise în secțiunea WORKING-STORAGE, iar citirea valorilor lor este realizată printr-o instrucțiune ACCEPT. Formatul general, simplificat, al acestei instrucțiuni este:

**ACCEPT** *nume-dată*

În această formă, instrucțiunea ACCEPT descrie operația de înregistrare, în zona asociată datei *nume-dată*, începând cu locația din stînga, a caracterelor perforate pe cartele începînd din prima coloană.

Trei aspecte sînt de menționat în ceea ce privește utilizarea instrucțiunii ACCEPT.

- În cazul în care pe o cartelă există mai multe date, ale căror valori trebuie citite prin instrucțiunea ACCEPT, ele trebuie grupate într-un articol, iar în instrucțiunea ACCEPT trebuie citat numele acestuia.

- În cazul în care prin instrucțiunea ACCEPT sînt citite mai multe cartele (un pachet de cartele), sfîrșitul acestora trebuie detectat prin program. De obicei, pentru această se poate proceda fie introducînd după ultima cartelă ce conține date de prelucrat o cartelă care, să conțină o valoare *v*, fie înregistrînd această valoare pe ultima cartelă ce conține date de prelucrat (dacă această cartelă nu este ocupată în întregime); indiferent de metoda folosită, după fiecare citire trebuie testată valoarea *v*.

- În general, instrucțiunea ACCEPT este utilizată pentru citirea unui volum mic de date.

- *Instrucțiunea ADD* descrie operația de adunare.

Format general 1:

$$\text{ADD} \left\{ \begin{array}{l} \text{nume-dată-1} \\ \text{literal-1} \end{array} \right\} \left[ , \left\{ \begin{array}{l} \text{nume-dată-2} \\ \text{literal-2} \end{array} \right\} \right] \dots$$

TO *nume-dată-m* [, *nume-dată-n*] ...

[ROUNDED]; [ON SIZE ERROR *instrucțiune*.]

Format general 2:

$$\text{ADD} \left\{ \begin{array}{l} \text{nume-dată-1} \\ \text{literal-1} \end{array} \right\} \left[ , \left\{ \begin{array}{l} \text{nume-dată-2} \\ \text{literal-2} \end{array} \right\} \right] \dots \text{GIVING } \text{nume-dată-n}$$

[ROUNDED] [;ON SIZE ERROR *instrucțiune*.]

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
9	0	7	0	9	0	5	5	9	9	9	9	9	7	7	

Fig. XII.15



Format general 3:

ADD { CORRESPONDING  
CORR } *nume-dată-1* TO *nume-dată-2*  
[ROUNDED] [;ON SIZE ERROR *instrucțiune*.]

Principiul de execuție este următorul:

- *format 1*: la valoarea fiecărei date *nume-dată-m*, *nume-dată-n* ... se adună valorile datelor *nume-dată-1/literal-1*, *nume-dată-2/literal-2* ...
- *format 2*: valorile datelor *nume-dată-1/literal-1*, *nume-dată-2/literal-2* ... sînt însumate, iar valoarea sumei este atribuită datei *nume-dată-n* (care poate fi definită și ca dată numerică de editare);
- *format 3*: valorile datelor corespondente ce aparțin datelor grupate *nume-dată-1* și *nume-dată-2* sînt adunate, iar valorile sumelor obținute sînt atribuite datelor corespondente ce aparțin datei grupate *nume-dată-2*.

#### Observație

Se numesc date corespondente, datele elementare care aparțin unor date grupate, au același nume și sînt de aceeași categorie. În exemplul din figura XII.16 datele corespondente sînt A,B și E. Considerînd datele din această figură, în urma execuției instrucțiunii:

ADD CORR ART-1 TO ART-2

valorile datelor A,B și E ce aparțin articolului ART-2 vor fi 10, 75 și, respectiv, 30.

*Instrucțiunea SUBTRACT* descrie operația de scădere.

Format general 1:

```
01 ART-1.
   05 A    PIC 9(4) VALUE 5.
   05 B    PIC 9(4) VALUE 25.
   05 C    PIC 99.
   05 D    PIC X(7).
   05 E    PIC 9(5) VALUE 10.
01 ART-2.
   05 A    PIC 9(6) VALUE 5.
   05 B    PIC 9(5) VALUE 50.
   05 C    PIC X(15).
   05 D    PIC 9(4).
   05 E    PIC 9(7) VALUE 20.
```

Fig. XII.16

SUBTRACT { *nume-dată-1*  
*literal-1* } [ , { *nume-dată-2*  
*literal-2* } ] ... FROM *nume-dată-n*  
[ROUNDED] [;ON SIZE ERROR *instrucțiune*.]

Format general 2:

SUBTRACT { *nume-dată-1*  
*literal-1* } [ , { *nume-dată-2*  
*literal-2* } ] ... FROM { *nume-dată-m*  
*literal-m* }  
GIVING *nume-dată-n*  
[ROUNDED] [;ON SIZE ERROR *instrucțiune*.]



Format general 3:

**SUBTRACT**  $\left\{ \begin{array}{l} \text{CORRESPONDING} \\ \text{CORR} \end{array} \right\} \text{nume-dată-1 FROM nume-dată-2}$   
[**ROUNDED**] [;ON **SIZE ERROR** instrucțiune.]

Principiul de execuție este următorul:

— *format 1*: valorile datelor *nume-dată-1/literal-1*, *nume-dată-2/literal-2* sînt însumate, suma obținută este scăzută din valoarea datei *nume-dată-n*, iar valoarea rezultatului este atribuită datei *nume-dată-n*;

— *format 2*: valorile datelor *nume-dată-1/literal*, *nume-dată-2/literal-2* ... sînt însumate, suma obținută este scăzută din valoarea datei *nume-dată-m/literal-m*, iar valoarea rezultatului este atribuită datei *nume-dată-n* (care poate fi definită și ca dată numerică de editare);

— *format 3*: valorile datelor corespondente ce aparțin datei grupate *nume-dată-1* sînt scăzute din valorile datelor corespondente ce aparțin datei grupate *nume-dată-2*, iar valorile rezultatelor sînt atribuite datelor corespondente ce aparțin datei grupate *nume-dată-2*.

Instrucțiunea **DIVIDE** descrie operația de împărțire.

Format general 1:

**DIVIDE**  $\left\{ \begin{array}{l} \text{nume-dată-1} \\ \text{literal-1} \end{array} \right\} \text{INTO nume-dată-2}$   
[**ROUNDED**] [;ON **SIZE ERROR** instrucțiune.]

Format general 2:

**DIVIDE**  $\left\{ \begin{array}{l} \text{nume-dată-1} \\ \text{literal-1} \end{array} \right\} \text{INTO } \left\{ \begin{array}{l} \text{nume-dată-2} \\ \text{literal-2} \end{array} \right\} \text{GIVING nume-dată-3}$   
[**REMAINDER** *nume-dată-4*]  
[**ROUNDED**] [;ON **SIZE ERROR** instrucțiune.]

Format general 3:

**DIVIDE**  $\left\{ \begin{array}{l} \text{nume-dată-1} \\ \text{literal-1} \end{array} \right\} \text{BY } \left\{ \begin{array}{l} \text{nume-dată-2} \\ \text{literal-2} \end{array} \right\} \text{GIVING nume-dată-3}$   
[**REMAINDER** *nume-dată-4*]  
[**ROUNDED**]; [ON **SIZE ERROR** instrucțiune.]

Principiul de execuție este următorul:

— *format 1*: valoarea datei *nume-dată-2* este împărțită la valoarea datei *nume-dată-1/literal-1*, iar valoarea cîtului este atribuită datei *nume-dată-2*;

— *format 2*: valoarea datei *nume-dată-2/literal-2* este împărțită la valoarea datei *nume-dată-1/literal-1*, valoarea cîtului este atribuită datei *nume-dată-3*, iar valoarea restului, datei *nume-dată-4*.

— *format 3*: valoarea datei *nume-dată-1/literal-1* este împărțită la valoarea datei *nume-dată-2/literal-2*, valoarea cîtului este atribuită datei *nume-dată-3*, iar valoarea restului, datei *nume-dată-4*.

Observații

● Datele *nume-dată-3* și *nume-dată-4* pot fi definite și ca date numerice de editare.

● În limbajul COBOL, restul este definit ca rezultat al diferenței dintre deîmpărțit și produsul dintre partea întreagă a cîtului și împărțitor.



● În prezența clauzei **ROUNDED**, valoarea citului este rotunjită după determinarea valorii restului.

*Instrucțiunea* **MULTIPLY** descrie operația de înmulțire.

Format general 1:

**MULTIPLY**  $\left\{ \begin{array}{l} \text{nume-dată-1} \\ \text{literal-1} \end{array} \right\}$  **BY** *nume-dată-2*  
[**ROUNDED**] [;**ON SIZE ERROR** *instrucțiune.*]

Format general 2

**MULTIPLY**  $\left\{ \begin{array}{l} \text{nume-dată-1} \\ \text{literal-1} \end{array} \right\}$  **BY**  $\left\{ \begin{array}{l} \text{nume-data-2} \\ \text{literal-2} \end{array} \right\}$  **GIVING** *nume-dată-3*  
[**ROUNDED**] [;**ON SIZE ERROR** *instrucțiune.*]

Principiul de execuție este următorul:

— *format 1*: valoarea datei *nume-dată-1/literal-1* este înmulțită cu valoarea datei *nume-dată-2*, iar valoarea produsului este atribuită datei *nume-dată-2*;

— *format 2*: valoarea datei *nume-dată-1/literal-1* este înmulțită cu valoarea datei *nume-dată-2/literal-2*, iar valoarea produsului este atribuită datei *nume-dată-3* (care poate fi definită ca și dată numerică de editare).

În tabelul XII.6 se exemplifică câteva operații aritmetice descrise prin instrucțiunile **ADD**, **SUBTRACT**, **DIVIDE**, **MULTIPLY**.

TABELUL XII.6

Instrucțiune	Valoarea datei							
	Înainte				După			
	A	B	C	D	A	B	C	D
ADD A TO B	3	6			3	9		
ADD A, B TO C	1	4	7		1	4	12	
ADD A, B TO C, D	1	4	7	22	1	4	12	27
ADD A, B GIVING C	1	4	7		1	4	5	
SUBTRACT A FROM B	5	9			5	4		
SUBTRACT A, B FROM C	1	4	15		1	4	10	
SUBTRACT A FROM B GIVING C	1	4	15		1	4	3	
SUBTRACT A, B FROM C GIVING D	1	4	15	22	1	4	15	10
SUBTRACT 3 FROM A	12				9			
SUBTRACT A, B FROM 22 GIVING C	2	3	47		2	3	17	
MULTIPLY A BY B	2	4			2	8		
MULTIPLY A BY B GIVING C	2	4	17		2	4	8	
MULTIPLY 4 BY A	3				12			
MULTIPLY A BY 6 GIVING B	3	4			3	18		
DIVIDE A INTO B	3	15			3	5		
DIVIDE A INTO B GIVING C	3	15	4		2	15	5	
DIVIDE A BY B GIVING C	24	12	19		24	12	2	
DIVIDE 2 INTO A	18				9			
DIVIDE A BY 3 GIVING B	24	2			24	8		
DIVIDE A BY B GIVING C								
REMAINDER D	19	7	0	0	19	7	2	5



## PROBLEME

1. Considerînd următoarea descriere a articolului ART:

01 ART.

05 A PIC 9(4) VALUE 12.

05 B PIC S9(4)V99 VALUE-123.A.

05 C PIC X(3) VALUE 'A1'.

05 D PIC A(4) VALUE 'ABC'.

să se indice care va fi conținutul zonelor asociate datelor ce compun articolul, după compilare.

2. Considerînd articolul de la punctul 4 să se indice care va fi conținutul zonelor asociate datelor ce compun acest articol, după execuția următoarelor instrucțiuni:

MOVE ZERO TO A

MOVE 55.6 TO B

MOVE '+12' TO C

MOVE ALL '\*' TO D

3. Fie un fișier pe cartele ale cărui articole conțin următoarele date: cod magazie (3 caractere numerice), cod material (5 caractere numerice), denumire material (20 caractere alfabetice), preț unitar (5 caractere numerice, ultimele două reprezentînd partea zecimală), cantitate (5 caractere numerice). Să se scrie un program pentru tipărirea la imprimantă a unui rînd pentru fiecare articol al fișierului pe cartele, care are codul magaziei egal cu 03. Rîndul va conține valorile următoarelor date: codul materialului, denumirea, prețul și cantitatea; aceste valori vor fi separate prin cîte 6 spații, iar valoarea codului materialului va fi tipărită începînd din coloana 10.

4. Să se indice care sînt erorile de sintaxă în cazul următoarelor instrucțiuni:

ADD A TO B GIVING C

DIVIDE 15 INTO A

SUBTRACT CORR ARTI FROM ARTI

5. Se consideră următoarele descrieri ale datelor A și B:

05 A PIC 9(4).

05 B PIC 9(3).

Să se descrie datele R1, R2, R3, R4 și R5, astfel încît zonele ce le vor fi rezervate să permită înregistrarea valorilor rezultatelor următoarelor operații:

$R1 = A + B$

$R2 = A/B$

$R3 = A * B$

$R4 = A ** B$

$R5 = (A - B)^2 - \sqrt{A^2 - B^2}$

6. Să se scrie un program care să realizeze:

— citirea de pe o cartelă a trei numere reale a, b, și c;

— determinarea valorii expresiei  $c = a/b + c^3$  și tipărirea acestei valori la imprimantă.

7. Să se scrie un program care să determine pătratele și radicalii primelor 50 numere naturale. Valorile obținute vor fi tipărite în cadrul unui tabel.



## CAPITOLUL XIII

### DESCRIEREA STRUCTURILOR DE CONTROL DE BAZĂ ÎN LIMBAJUL COBOL

Scopul acestui capitol este de a prezenta ordinea în care operațiile unui program COBOL, descrise prin instrucțiuni, sînt efectuate, adică modul în care se transferă controlul de la o instrucțiune la alta, sub aspectul execuției programului. Structura implicată de această ordonare a execuției instrucțiunilor reprezintă *structura de control* a programului. Ea permite realizarea de *salturi* (ramificări) condiționate sau necondiționate, *cicluri* cu număr fix sau variabil de iterații, *reutilizarea* diferitelor părți ale programului etc. În cele ce urmează sînt prezentate trei structuri de control de bază — structuri *secvențiale*, structuri *alternative* și structuri *repetitive* (fig. XIII.1) — precum și extensiile lor.

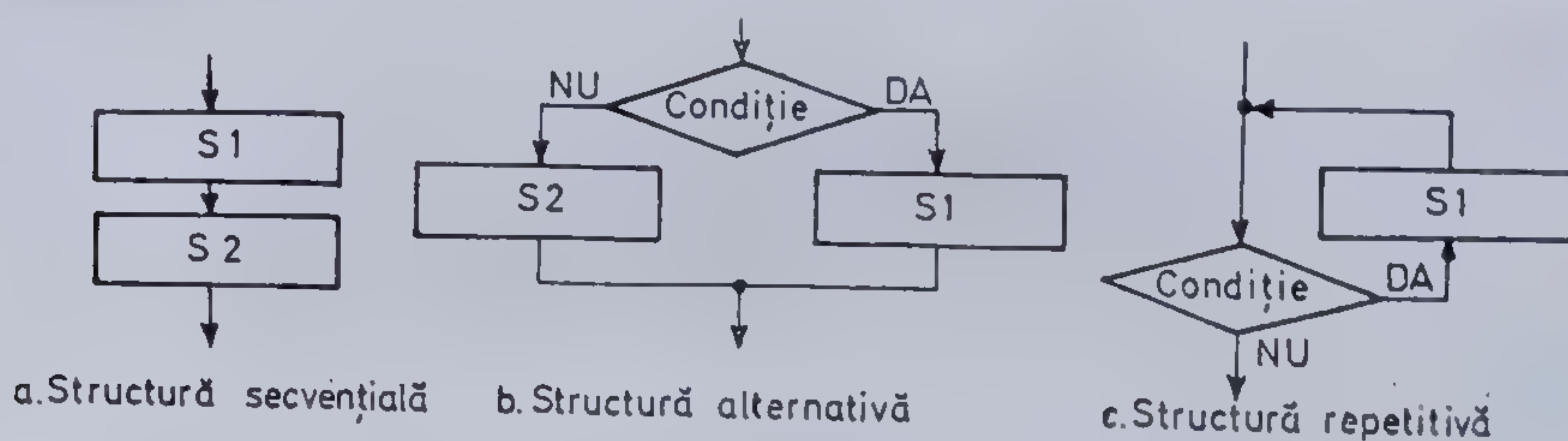


Fig XIII.1

Instrucțiunile care permit descrierea acestor structuri sînt: **PERFORM**, **GO TO**, **EXIT**, **IF**, **GO TO DEPENDING**, **PERFORM TIMES** și **PERFORM UNTIL**.

#### 1. STRUCTURI SECVENȚIALE

Aceste structuri se descriu prin intermediul instrucțiunii **PERFORM**, al cărei format general este următorul:

**PERFORM** *nume-prelucrare-1* [**THRU** *nume-prelucrare-2*]

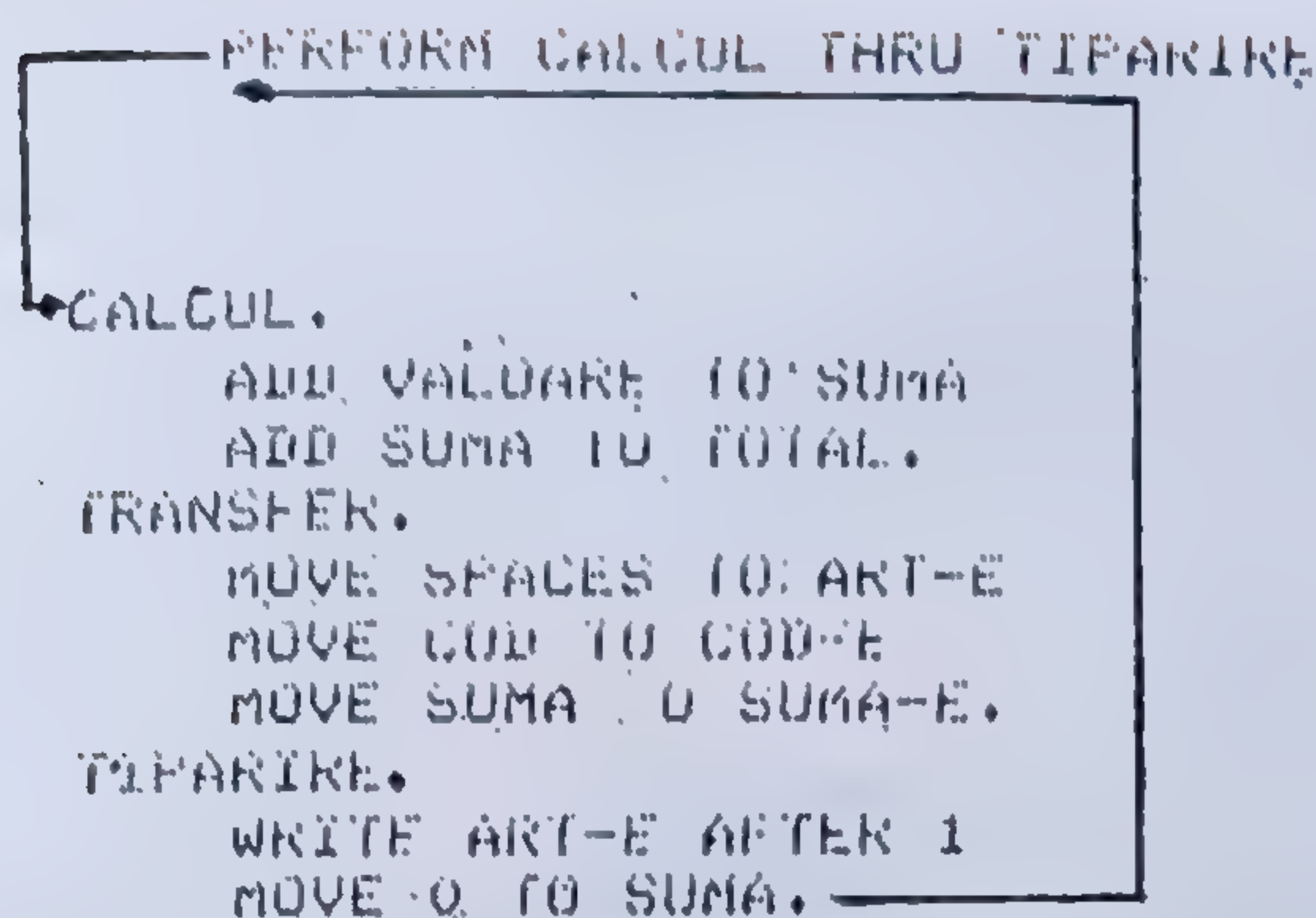
Utilizarea instrucțiunii **PERFORM** impune gruparea instrucțiunilor programului în paragrafe și, eventual, a paragrafelor în secțiuni și identificarea acestora prin etichete (nume de prelucrare), în vederea referirii.

Instrucțiunea **PERFORM** descrie, în ordine, următoarele operații:

- saltul la o secvență de prelucrare, care reprezintă un singur paragraf (secțiune) sau un grup de paragrafe (secțiuni) succesive;
- execuția acestei secvențe;
- saltul (revenirea) la instrucțiunea ce urmează instrucțiunii **PERFORM**.

În figura XIII.2 este prezentat principiul de execuție al instrucțiunii **PERFORM**, în cazul în care secvența apelată reprezintă o succesiune de paragrafe (primul fiind **CALCUL**, iar ultimul **TIPARIRE**).





b.

Fig. XIII.2

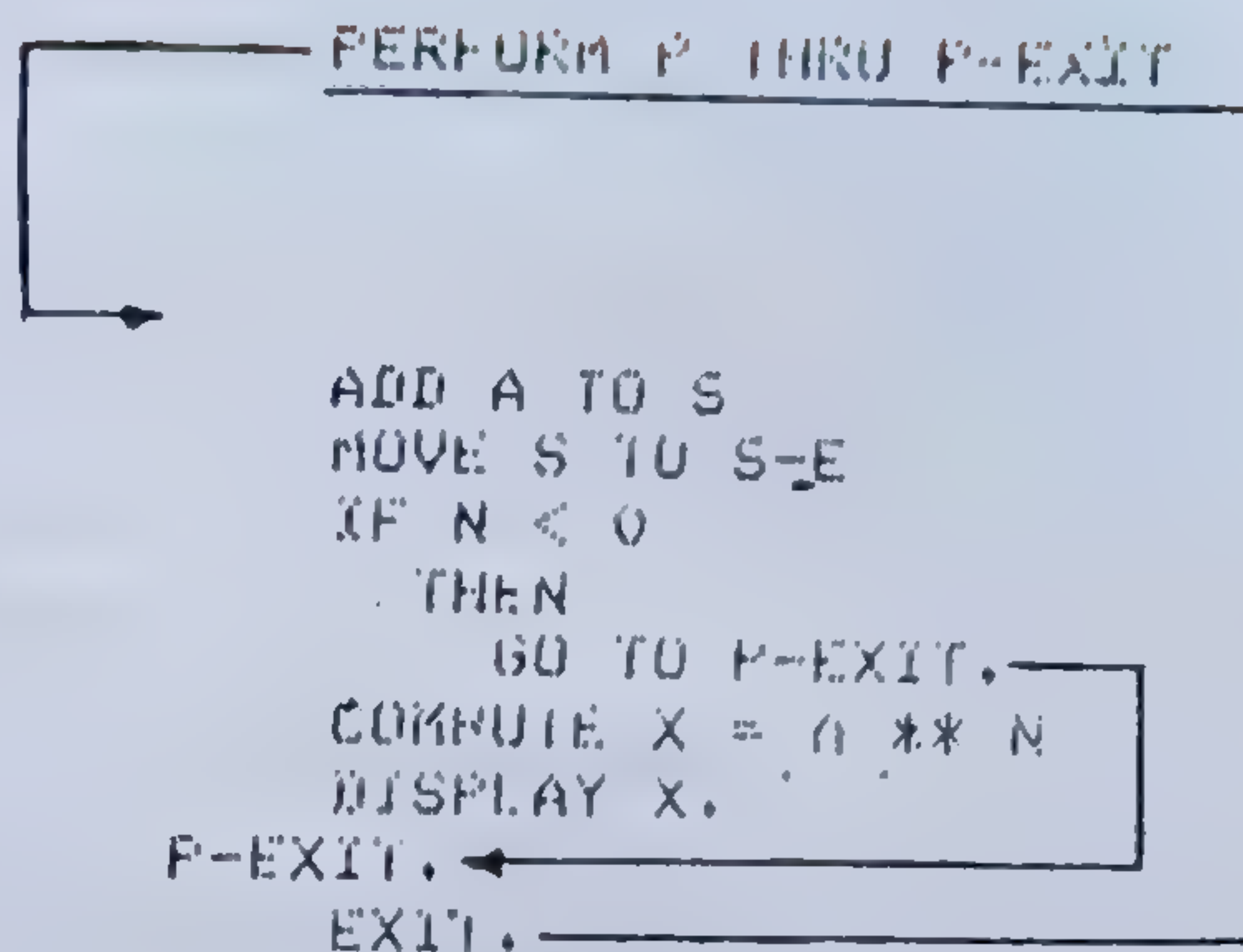


Fig. XIII.3

### Observații

● Paragrafele (secțiunile) apelate prin instrucțiunea **PERFORM** nu trebuie, în general, să urmeze (fizic) acesteia.

● În cazul utilizării formei de scriere:

**PERFORM** *nume-prelucrare-1 THRU* *nume-prelucrare-2.*

deoarece paragrafele (secțiunile) ce constituie secvența apelată trebuie să fie grupate, logica programului este dependentă de plasarea fizică a acestor paragrafe (secțiuni). Din acest motiv, este indicată utilizarea instrucțiunii **PERFORM** fără opțiunea **THRU**.

Totuși, există situații care impun folosirea opțiunii **THRU**; de exemplu, atunci când prin testarea unei condiții, în cadrul unui paragraf **P** apelat prin **PERFORM**, se decide abandonarea execuției instrucțiunilor ce urmează celei care descrie operația de testare. Realizarea acestui lucru necesită asocierea paragrafului **P**, a unui paragraf care să conțină o instrucțiune **EXIT** și plasarea, în instrucțiunea de testare din paragraful **P**, a unei instrucțiuni **GO TO**, în care trebuie citat numele paragrafului ce urmează lui **P** (fig. XIII.3).

**Instrucțiunea GO TO**, al cărei format general este:

**GO TO** *nume-prelucrare*

descrie operația de salt necondiționat la prima instrucțiune a paragrafului (secțiunii) *nume-prelucrare*.

**Instrucțiunea EXIT** al cărei format general este:

*nume-paragraf* **EXIT.**

descrie operația de revenire la instrucțiunea ce urmează instrucțiunii **PERFORM** sau, cu alte cuvinte, operația de ieșire dintr-o secvență apelată prin instrucțiunea **PERFORM**.

*Exemplu.* Fie un fișier pe cartele **FIS-PRODUCTII**, ale cărui articole conțin date referitoare la valoarea producției realizată de întreprinderile unei centrale industriale (fig. XIII.4). Articolele sînt ordonate crescător, după valorile datei **COD-CENTRALA**. Se consideră că o centrală are în subordine mai multe întreprinderi. Să se scrie un program care să editeze un raport de forma celui prezentat în figura XIII.5. Fiecare rînd al raportului va conține codul unei centrale și valoarea producției realizată de întreprinderile acesteia.

O modalitate de rezolvare a acestei probleme o reprezintă programul XIII.1 (vezi schema logică din figura XIII.6). Programul reflectă utilizările

### PRODUCTIE

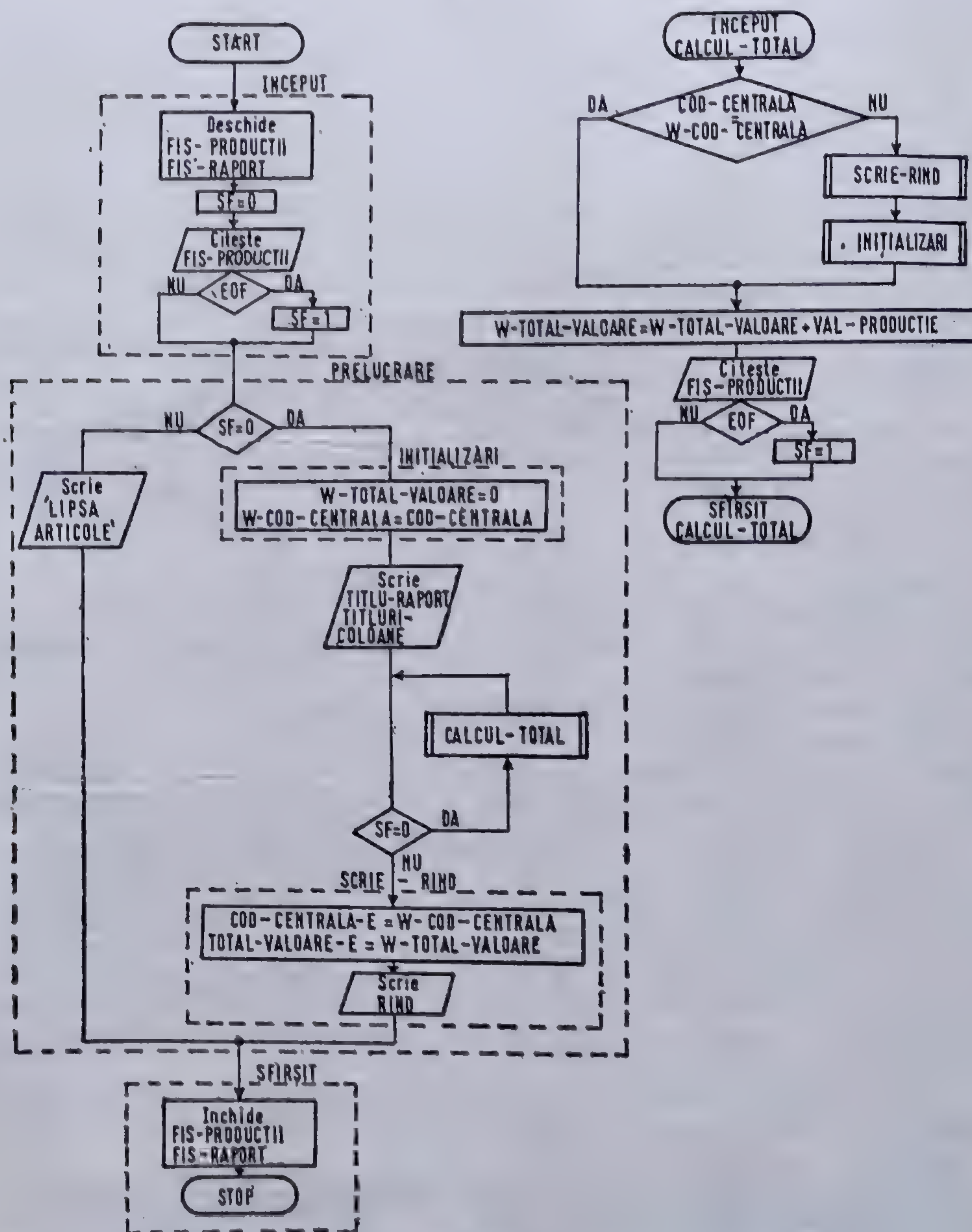
COD-MINISTER	COD-CENTRALA	COD-INTREPRINDERE	VAL-PRODUCTIE
9(2)	9(2)	9(3)	9(8)V99

Fig. XIII.4



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41		
																			</																							

**Fig. XIII.5**



**Fig. XIII.6**



cele mai frecvente ale instrucțiunii PERFORM: gruparea instrucțiunilor în paragrafe pentru a pune în evidență componentele funcționale ale programului și reutilizarea unor părți ale acestuia; reutilizarea constă în: scrierea o singură dată a unei secvențe de instrucțiuni care trebuie să apară de mai multe ori în program; plasarea acestei secvențe într-un loc din program, încât să nu afecteze ordinea execuției instrucțiunilor; apelarea acestei secvențe pentru execuție, prin instrucțiunea PERFORM, din locurile în care secvența ar fi trebuit inserată.

## 2. STRUCTURI ALTERNATIVE

Aceste structuri se descriu prin intermediul instrucțiunii IF, care specifică ordinea de execuție a instrucțiunilor, în funcție de valoarea logică a unei condiții.

Format general:

IF *condiție* THEN {*instrucțiune-1*  
NEXT SENTENCE}  
ELSE {*instrucțiune-1*  
NEXT SENTENCE}

unde:

— *condiție* reprezintă o construcție COBOL, numită *test*, prin intermediul căreia se poate verifica: relația dintre termeni (test de relație), categoria unei date (test de clasă), valoarea algebrică a unei date numerice (test de semn), dacă valoarea unei date aparține unei mulțimi de valori (test de nume de condiție);

— *instrucțiune-1* și *instrucțiune-2* reprezintă, fiecare, una sau mai multe instrucțiuni COBOL, care vor fi numite, în continuare, *secvențe*.

În general, o instrucțiune IF descrie următoarele operații:

- stabilirea valorii logice a condiției („adevărat” sau „fals”);
- execuția secvenței desemnate prin *instrucțiune-1*, dacă valoarea logică a condiției este „adevărat”;
- execuția secvenței desemnate prin *instrucțiune-2*, dacă valoarea logică a condiției este „fals”;

— saltul la prima instrucțiune a frazei următoare, dacă instrucțiunea conține opțiunea *NEXT SENTENCE* sau dacă ia sfârșit execuția secvenței indicate prin *instrucțiune-1* sau *instrucțiune-2*.

● O generalizare a structurilor de control alternative, descrise prin instrucțiunea IF, este cea prezentată în figura XIII.7, numită *structură de*

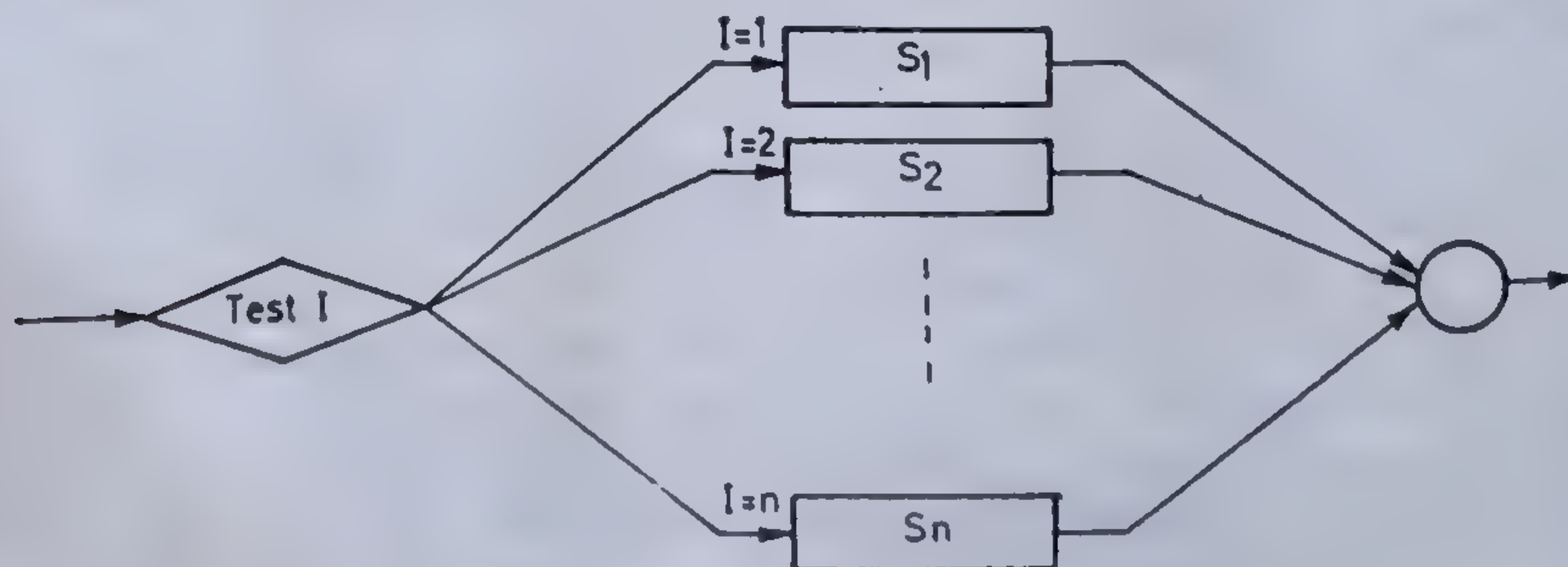


Fig. XIII.7



control alternativă generalizată. În figura XIII.7,  $I$  reprezintă o dată numerică, cu valori cuprinse între 1 și  $n$ , iar  $S_i$  ( $i = 1, 2, \dots, n$ ) secvențele de instrucțiuni ce vor fi executate în funcție de valoarea lui  $I$ , stabilită prin test.

Limbajul COBOL nu dispune de instrucțiuni elementare pentru descrierea acestor tipuri de structuri de control. Totuși, ele pot fi simulate utilizând:  
a) instrucțiuni IF sau b) instrucțiunea GO TO DEPENDING.

*Exemplu:*

Se consideră că o cartelă conține trei date: TIP-EXPRESIE, A și B.

Valorile datei TIP-EXPRESIE — care pot fi 1, 2 sau 3 — indică determinarea valorii unei expresii aritmetice și anume:

- $E1 = \sqrt{A - B}$ , dacă TIP-EXPRESIE = 1;
- $E2 = \sqrt{A} - \sqrt{B}$ , dacă TIP-EXPRESIE = 2;
- $E3 = \sqrt[3]{A + B}$ , dacă TIP-EXPRESIE = 3.

Se cere să se scrie un program care să determine și să tipărească, prin instrucțiunea DISPLAY, valoarea expresiei indicată de valoarea datei TIP-EXPRESIE.

O modalitate de rezolvare, utilizând instrucțiunea IF pentru testarea valorii datei TIP-EXPRESIE, este dată în programul XIII.2.

Structura de control descrisă prin cele trei instrucțiuni IF are o singură intrare și o singură ieșire indiferent de valoarea datei TIP-EXPRESIE. De exemplu, dacă această valoare este 2, va fi executat paragraful CALCUL-E 2 după care execuția programului va continua cu execuția instrucțiunii STOP. Secvența EROARE a fost prevăzută pentru tratarea situației în care valorile datei TIP-EXPRESIE nu sînt cuprinse între 1 și 3.

O altă modalitate de rezolvare constă în utilizarea instrucțiunii GO TO DEPENDING al cărei format este următorul:

**GO TO** *nume-prelucrare-1* [, *nume-prelucrare-2*] ...  
**DEPENDING ON** *nume-dată*

unde:

— *nume-prelucrare-1*, *nume-prelucrare-2* ... reprezintă nume de paragrafe sau secțiuni;

— *nume-dată* reprezintă o dată numerică, ale cărei valori sînt numere întregi și pozitive.

Această instrucțiune descrie un salt la prima instrucțiune a paragrafului (secțiunii) *nume-prelucrare-1*, *nume-prelucrare-2*, ... după cum valoarea datei *nume-dată* este egală cu 1, 2, ... Dacă această valoare nu este cuprinsă între 1 și  $n$  (unde  $n$  reprezintă numărul numelor de prelucrare indicate în instrucțiune), execuția programului continuă cu instrucțiunea ce urmează lui GO TO. Pentru prevenirea erorilor care pot apărea la execuția programului instrucțiunea GO TO DEPENDING trebuie să fie urmată de o secvență pentru tratarea situației în care valorile datei *nume-dată* nu corespund numărului numelor de prelucrare. De asemenea, o altă observație care se poate face este aceea că, în instrucțiunea GO TO DEPENDING, un nume de prelucrare poate apărea de mai multe ori.



```

ID DIVISION.
PROGRAM-ID. PIII2.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 CARTELA.
    05 TIP-EXPRESIE PIC 9.
    05 A PIC S99V99.
    05 B PIC S99V9.
01 VALORI-EXPRESII.
    05 E1 PIC +9(5).99.
    05 E2 PIC +9(5).99.
    05 E3 PIC +9(7).99.
PROCEDURE DIVISION.
START.
    ACCEPT CARTELA
    IF TIP-EXPRESIE = 1
        THEN
            PERFORM CALCUL-E1
        ELSE
            IF TIP-EXPRESIE = 2
                THEN
                    PERFORM CALCUL-E2
                ELSE
                    IF TIP-EXPRESIE = 3
                        THEN
                            PERFORM CALCUL-E3
                        ELSE
                            PERFORM EROARE.

    STOP RUN.
CALCUL-E1.
    COMPUTE E1 = (A - B) ** (1 / 2)
    DISPLAY 'E1 = ' E1.
CALCUL-E2.
    COMPUTE E2 = A ** 0.5 - B ** 0.5
    DISPLAY 'E2 = ' E2.
CALCUL-E3.
    COMPUTE E3 = (A + B) ** (1 / 3)
    DISPLAY 'E3 = ' E3.
EROARE.
    DISPLAY '*** VALOAREA DATEI TIP-EXPRESIE ERONATA ***'.

```

#### PROGRAMUL XIII.2

Transcrierea în limbajul COBOL, utilizând instrucțiunea GO TO DEPENDING, a schemei programului XIII.2 se poate realiza astfel:

```

START.
    ACCEPT CARTELA
    PERFORM PRELUCRARE THRU PRELUCRARE-EXIT
    STOP RUN.
PRELUCRARE.
    GO TO CALCUL-E1
        CALCUL-E2
        CALCUL-E3
        DEPENDING ON TIP-EXPRESIE.
    GO TO EROARE.
CALCUL-E1.

```



```

    COMPUTE E1 = (A - B) ** 0.5
    DISPLAY' E1 = ' E1
    GO TO PRELUCRARE-EXIT.
CALCUL-E2.
    COMPUTE E2 = A ** (1 / 2) - B ** (1 / 2)
    DISPLAY' E2 = ' E2
    GO TO PRELUCRARE-EXIT.
CALCUL-E3.
    COMPUTE E3 = (A + B) ** (1 / 3)
    DISPLAY' E3 = ' E3
    GO TO PRELUCRARE-EXIT.
EROARE.
    DISPLAY '*** VALORILE DATEI'
    'TIP-EXPRESIE NU SÎNT CUPRINSE'
    'ÎNTRE 1 ȘI 3 ***'.
PRELUCRARE-EXIT.
EXIT.

```

### 3. STRUCTURI REPETITIVE

Aceste structuri sînt numite și cicluri; *ciclul* reprezintă cea mai simplă modalitate de a executa o secvență de instrucțiuni de mai multe ori. O execuție a acestei secvențe reprezintă o *iterație* a ciclului, iar numărul total de execuții, *numărul de iterații* ale ciclului. Din acest punct de vedere există cicluri cu număr fix de iterații și cicluri cu număr variabil de iterații.

Pentru specificarea completă a unui ciclu, instrucțiunea care îl descrie trebuie să indice:

- structura ciclului, adică instrucțiunile care aparțin ciclului (corpul ciclului);

- controlul ciclului, adică modalitatea de a controla iterațiile.

În limbajul COBOL corpul unui ciclu reprezintă o secvență de instrucțiuni ce constituie un paragraf (secțiune) sau un grup de paragrafe (secțiuni) succesive, fiind indicat în instrucțiunea PERFORM prin identificatorul *nume-prelucrare-1* și, respectiv, prin construcția *nume-prelucrare-1 THRU nume-prelucrare-2*.

a. **Cicluri cu număr fix de iterații.** Pentru descrierea acestor cicluri este utilizată instrucțiunea PERFORM TIMES, al cărei format general este următorul:

**PERFORM** *nume-prelucrare-1* [THRU *nume-prelucrare-2*]

$$\left. \begin{array}{l} \textit{literal} \\ \textit{nume-dată} \end{array} \right\} \text{TIMES}$$

unde: *literal* reprezintă un literal numeric, iar *nume-dată*, o dată numerică. Valoarea literalului sau datei trebuie să fie un număr întreg pozitiv, diferit de zero. Considerînd că această valoare este *n*, corpul ciclului va fi executat de *n* ori. În cazul în care  $n \leq 0$ , instrucțiunea PERFORM este inefectivă. Trebuie făcută observația că dacă numărul de iterații este indicat de valoarea datei *nume-dată*, aceasta poate fi utilizată în corpul ciclului, fără ca acest fapt să influențeze controlul ciclului.



```

ID DIVISION.
PROGRAM-ID. PIII3.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 N          PIC 999.
01 NUMAR-SUMA.
   05 NUMAR PIC 999 VALUE 1
   05 SUMA  PIC 9(5) VALUE ZERO.
PROCEDURE DIVISION.
START
  ACCEPT N
  PERFORM ADUNA N TIMES.
  DISPLAY 'SUMA PRIMELOR.' N NUMERE NATURALE : SUMA
  STOP RUN.

ADUNA.
  ADD NUMAR TO SUMA
  ADD 1 TO NUMAR.

```

#### PROGRAMUL XIII.3

O exemplificare a utilizării instrucțiunii PERFORM TIMES este prezentată în programul XIII.3 care determină și afișează la imprimantă, prin instrucțiunea DISPLAY, suma primelor N numere naturale; N este citit de pe o cartelă, prin intermediul instrucțiunii ACCEPT (schema logică din figura XIII.8).

b. **Cicluri cu un număr variabil de iterații.** Caracteristic acestor cicluri este faptul că pentru determinarea momentului transferului controlului în afara ciclului (ieșirea din ciclu) este necesară evaluarea unei condiții, care se poate face înainte sau după execuția corpului ciclului (fig. XIII.9); ciclul din figura III.9 a, se numește *structură de control repetitivă condiționată anterior*, iar cel din figura XIII.9 b, *structură de control repetitivă condiționată posterior*. Deosebirea dintre cele două tipuri de cicluri constă în faptul că primul poate să nu aibă nici o iterație, pe când cel de-al doilea are cel puțin o iterație.

● O structură repetitivă condiționată anterior, de tipul celei din figura XIII.10 b, poate fi descrisă prin intermediul instrucțiunii PERFORM UNTIL, al cărei format general este următorul:

**PERFORM** *nume-prelucrare-1* [THRU *nume-prelucrare-2*] **UNTIL** *condiție*  
unde *condiție* reprezintă o condiție de tipul celor admise de instrucțiunea IF.

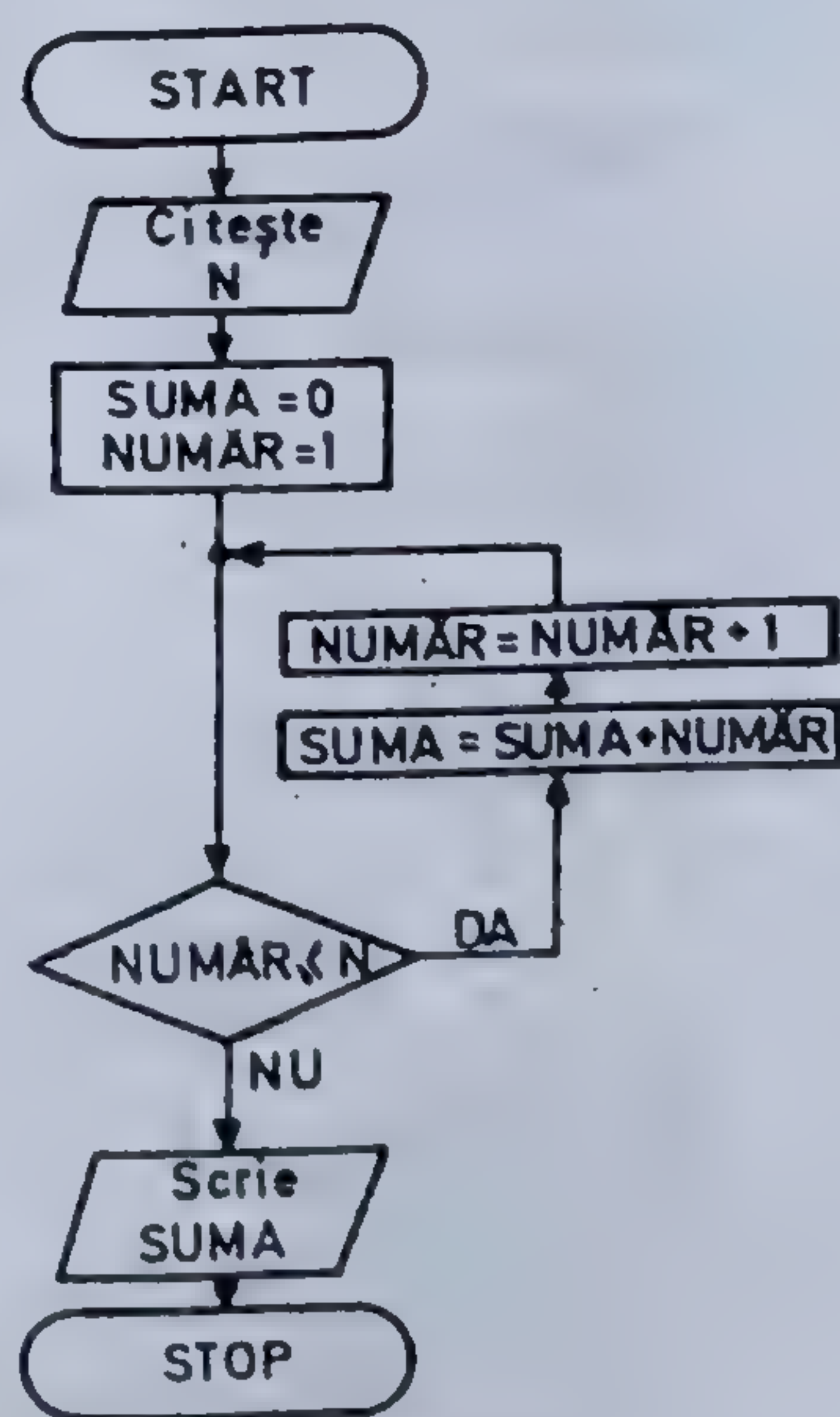
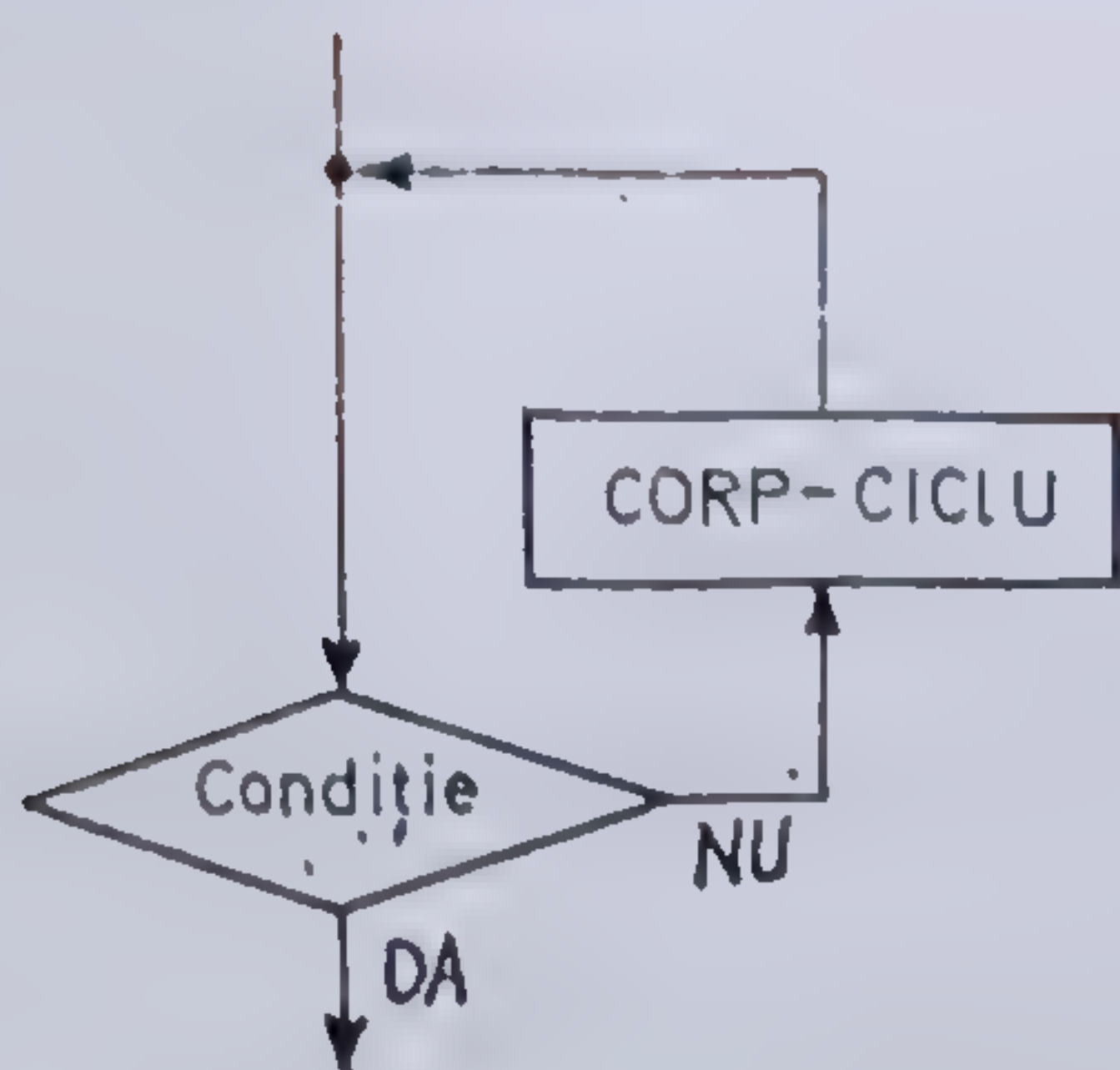
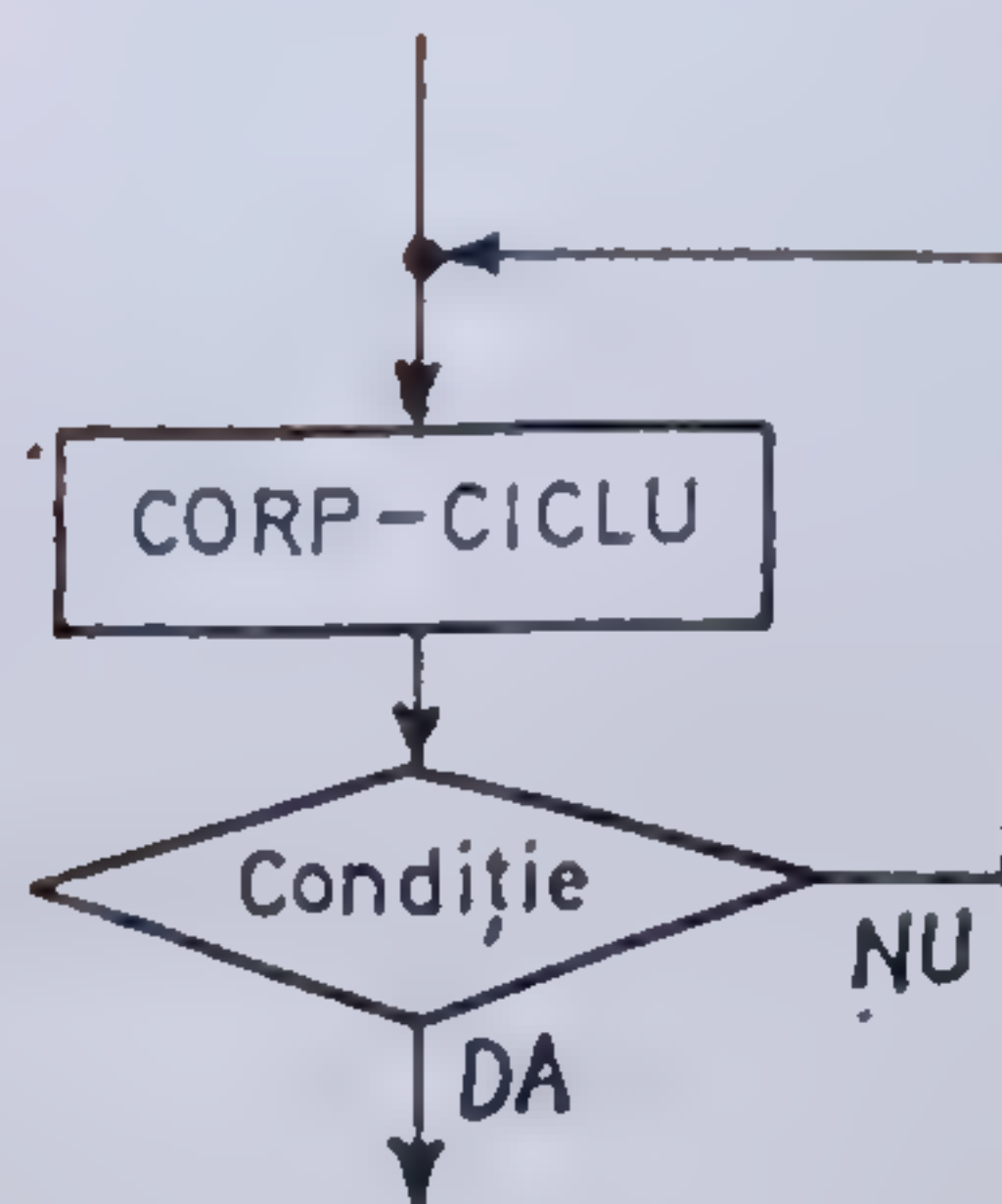


Fig. XIII.8





a.



b.

Fig. XIII.9

*Exemplu.* Prelucrarea din figura XIII.8 poate fi descrisă și astfel:

START.

ACCEPT N

PERFORM ADUNA

UNTIL NUMAR > N

DISPLAY 'SUMA PRIMELOR' N

'NUMARE NATURALE: ' SUMA

STOP RUN.

ADUNA.

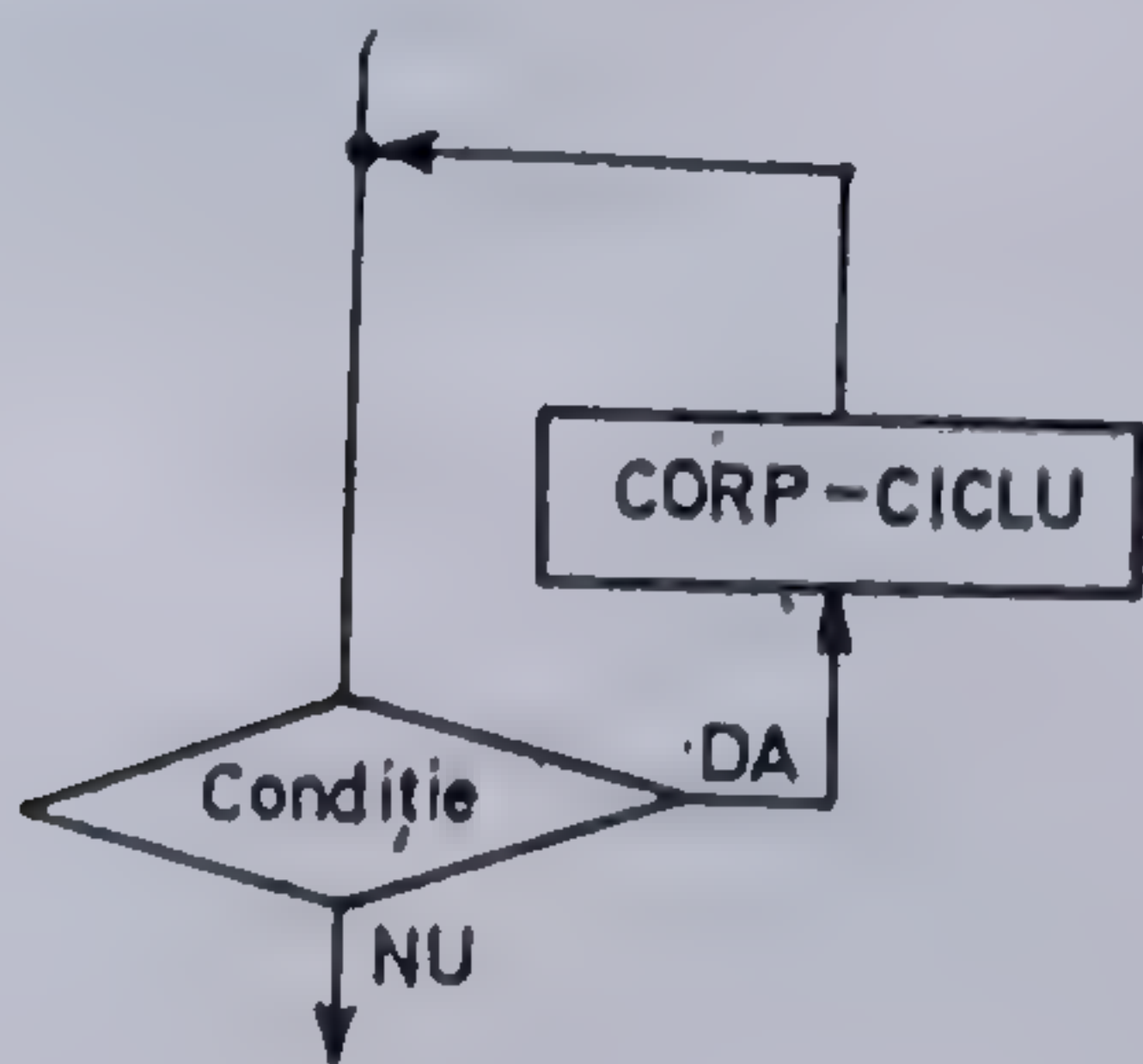
ADD NUMAR TO SUMA

ADD 1 TO NUMAR.

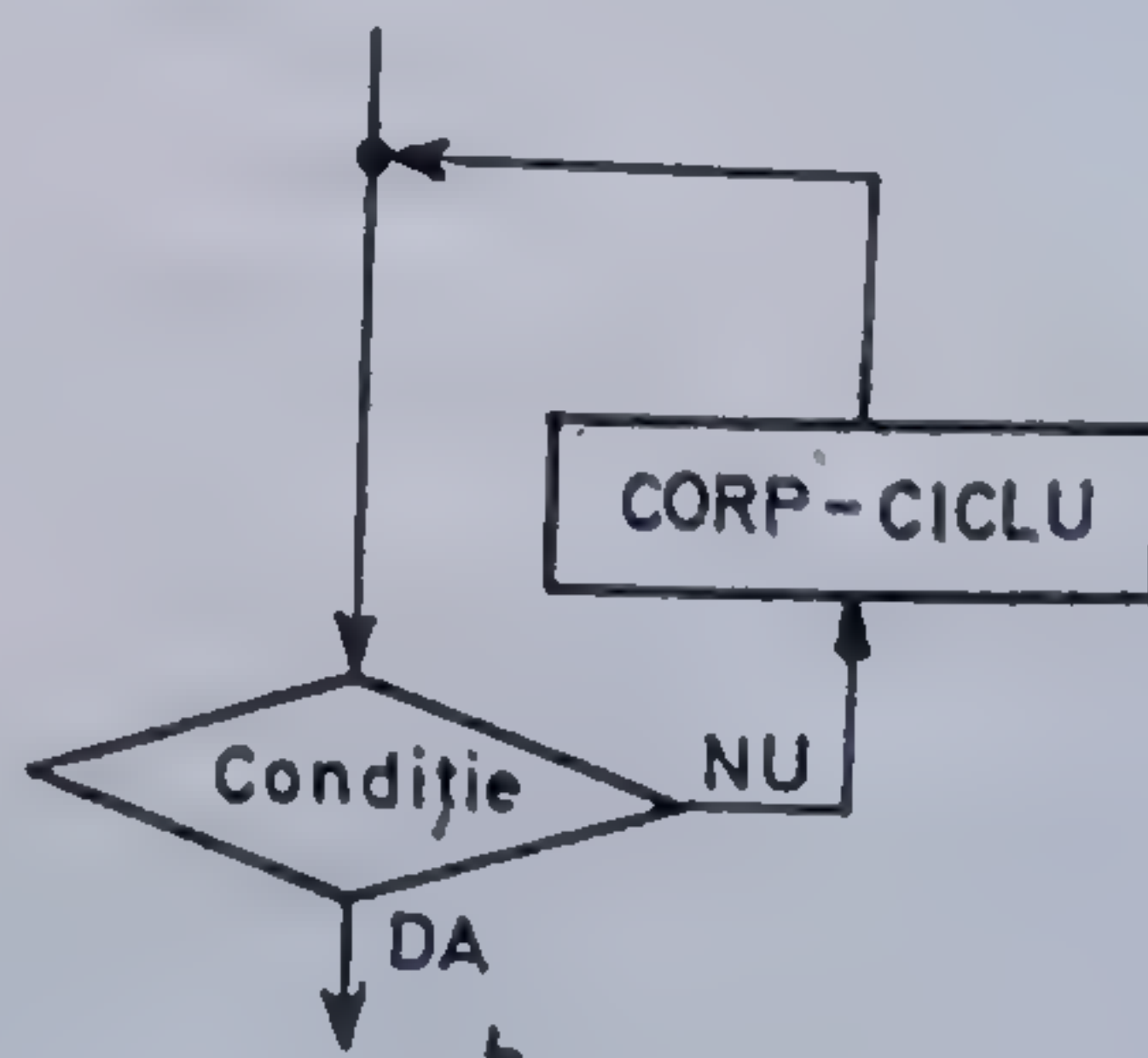
Deoarece corpul ciclului este executat atîta timp cît valoarea logică a condiției testate este „fals” o structură repetitivă condiționată anterior de tipul celei prezentate în figura XIII.10, a, poate fi descrisă, tot printr-o instrucțiune PERFORM UNTIL, însă negînd condiția:

**PERFORM** *nume-prelucrare-1* [THRU *nume-prelucrare-2*]  
**UNTIL NOT** (*condiție*)

• Limbajul COBOL nu dispune de instrucțiuni elementare pentru descrierea structurilor de control repetitive condiționate posterior, de tipul celei din figura XIII.9 b. Ele pot fi însă simulate prin două instrucțiuni:



a.



b.

Fig. XIII.10



PERFORM și PERFORM UNTIL. De exemplu, schema din figura XIII.8 poate fi transcrisă astfel:

```
START.  
  ACCEPT N  
  PERFORM ADUNA  
  PERFORM ADUNA  
    UNTIL NUMAR > N  
  DISPLAY 'SUMA PRIMELOR' N  
    'NUMERE NATURALE: ' SUMA  
  STOP RUN  
ADUNA.  
  ADD NUMAR TO SUMA  
  ADD 1 TO NUMAR.
```

#### Observație

Ciclurile condiționate anterior sunt preferabile celor condiționate posterior intrucit asigură un control mai riguros al iterațiilor.

#### PROBLEME

1. Să se scrie un program pentru rezolvarea ecuației de gradul 2. Coeficienții ecuației vor fi citați de pe o cartelă, iar soluțiile vor fi tipărite la imprimantă.
2. Să se scrie un program, în două variante, care să determine și să tipărească la imprimantă produsul primelor  $N$  numere naturale, utilizând instrucțiunea PERFORM TIMES și, respectiv, PERFORM UNTIL.
3. Se consideră un fișier pe cartele ale cărui articole conțin date referitoare la situația școlară a elevilor unui liceu, și anume: clasa, nume, media generală anuală. Să se scrie un program care să determine și să afișeze la imprimantă media generală pentru fiecare clasă precum și numele elevului cu cea mai mare medie anuală. Se va considera că articolele fișierului pe cartele sînt ordonate după valorile datei CLASA.



## ELEMENTE COMPLEMENTARE DE DESCRIERE ȘI PRELUCRARE A DATELOR

### 1. REPREZENTAREA INTERNĂ A DATELOR. CLAUZA USAGE

În funcție de caracterele ce compun valorile datelor, în limbajul COBOL pot fi definite patru categorii de date: alfabetice, alfanumerice, numerice de calcul și numerice de editare.

Valorile datelor sînt reprezentate în memoria centrală în zone rezervate de compilator, formatul de reprezentare fiind determinat de categoria datei. Lungimea zonelor-memorie rezervate datelor este funcție de lungimea datelor și de formatul de reprezentare internă a valorilor acestora și se exprimă în număr de locații-memorie.

a. **Reprezentarea datelor nenumeric.** Valorile datelor alfabetice, alfanumerice și numerice de editare sînt reprezentate în memorie în *cod EBCDIC*, adică fiecare caracter este reprezentat într-o locație prin codul intern EBCDIC care îi corespunde. Acest format de reprezentare se numește reprezentare în cod EBCDIC (sau reprezentare DISPLAY).

Lungimea zonei-memorie asociată unei date alfabetice, alfanumerice sau numerice de editare coincide cu lungimea datei și este stabilită de compilator în funcție de simbolurile care figurează în șablonul de descriere.

b. **Reprezentarea datelor numerice.** Valorile datelor numerice de calcul pot fi reprezentate în următoarele formate:

- zecimal-despachetat;
- zecimal-împachetat;
- cod complementar;
- virgulă mobilă simplă precizie;
- virgulă mobilă dublă precizie.

Aceste formate de reprezentare sînt indicate prin opțiunile *clauzei USAGE*, al cărei format general este următorul:

<div style="display: flex; align-items: center; justify-content: center;"> <div style="margin-right: 10px;">[USAGE IS]</div> <div style="text-align: center;"> <div style="margin-bottom: 5px;">DISPLAY</div> <div style="display: flex; align-items: center;"> <div style="margin-right: 5px;">{</div> <div style="text-align: center;"> COMPUTATIONAL-3 CPMP-3 COMPUTATIONAL COMP COMPUTATIONAL-1 COMP-1 COMPUTATIONAL-2 COMP-2 </div> <div style="margin-left: 5px;">}</div> </div> </div> </div>
--

● *Opțiunea DISPLAY* este implicită și indică formatul de reprezentare *zecimal-despachetat*. Caracterele (cifrele) ce compun valorile datelor sînt înregistrate fiecare într-o locație, prin codul EBCDIC care le corespunde; dacă valorile reprezintă numere cu semn, acesta se indică în șablonul de



descriere prin simbolul S. Lungimea zonei-memorie asociată datei este egală cu lungimea datei și nu poate depăși 18 locații. Aceasta înseamnă că în format zecimal-despachetat pot fi reprezentate valori cuprinse între  $-10^{18} + 1$  și  $10^{18} - 1$ .

În general, formatul de reprezentare zecimal-despachetat este utilizat pentru valorile datelor care se citesc de pe cartele, care se tipăresc la imprimantă etc.

● **Opțiunea COMPUTATIONAL-3** indică formatul de reprezentare *zecimal-împachetat*. Caracterele ce compun valorile datelor sînt înregistrate, codificate binar, cîte două într-o locație, cu excepția ultimei locații în care este înregistrat semnul și un caracter.

Lungimea zonei-memorie asociată datei se stabilește conform relației:

$$l_z = \left[ \frac{l_d}{2} \right] + 1$$

unde:  $l_z$  reprezintă lungimea zonei-memorie, iar  $l_d$ , lungimea datei. Lungimea maximă a zonei-memorie asociată datei este de 10 locații; în această zonă pot fi înregistrate valori cuprinse între  $-10^{20} + 1$  și  $10^{20} - 1$  (însă numai prin operații de atribuire, prin inițializare intervalul fiind  $[-10^{19} + 1, 10^{19} - 1]$ ).

Formatul de reprezentare zecimal-împachetat poate fi utilizat pentru înregistrarea valorilor numerice pe suporturi magnetice sau pentru valorile care reprezintă operanzi ai instrucțiunilor aritmetice.

● **Opțiunea COMPUTATIONAL** indică formatul de reprezentare în *cod complementar* (sau reprezentare binară), caracteristic valorilor ce reprezintă numere întregi pozitive sau negative.

Valorile datelor descrise cu această opțiune sînt înregistrate în zone-memorie de lungime  $l_z$  egale cu:

— 2 locații, dacă valoarea datei conține de la 1 pînă la 4 caractere (cifre);

— 4 locații, dacă valoarea datei conține de la 5 pînă la 9 caractere;

— 8 locații, dacă valoarea datei conține de la 10 pînă la 18 caractere.

Valorile datelor reprezentate în cod complementar sînt cuprinse în intervalul  $[-2^n, 2^n - 1]$  unde  $n = 15, 31$  sau 63, după cum 1 este egal cu 2, 4, sau, respectiv, 8 locații; compilatorul COBOL atribuie acestor zone adrese multiplu de 2,4 și respectiv, 8.

Formatul de reprezentare în cod complementar poate fi utilizat pentru înregistrarea valorilor numerice întregi pe suporturi magnetice, pentru reprezentarea valorilor indicilor, a valorilor operanzilor instrucțiunilor aritmetice etc. Avantajul acestui format de reprezentare constă în utilizarea eficientă a suporturilor externe, a memoriei, precum și în creșterea performanței programelor.

● **Opțiunile COMPUTATIONAL-1 și COMPUTATIONAL-2** indică formatul de reprezentare în *virgulă mobilă simplă precizie* și, respectiv, în *virgulă mobilă dublă precizie*, caracteristic valorilor ce reprezintă numere reale.

Datele descrise cu opțiunile COMPUTATIONAL-1 și COMPUTATIONAL-2 li se rezervă zone-memorie de lungime 4 și, respectiv, 8 locații, aliniate la adrese multiplu de 4 și, respectiv, 8.



În virgulă mobilă pot fi reprezentate valori cuprinse între  $-0,72 \cdot 10^{76}$  și  $0,72 \cdot 10^{76}$ .

Formatul de reprezentare în virgulă mobilă este utilizat pentru a putea prelucra date numerice a căror lungime depășește 18 caractere.

Intrucât această formă de reprezentare este standardizată (normalizată), lungimea datelor nu trebuie specificată prin clauza PICTURE.

#### Observație

Clauza USAGE poate figura atât în rubricile de descriere a datelor elementare cit și în rubricile de descriere a datelor grupate. În ultimul caz, ea se aplică tuturor datelor care alcătuiesc datele grupate.

## 2. ALINIAREA DATELOR. CLAUZA SYNCHRONIZED

Clauzele PICTURE și USAGE definesc complet, în limbajul COBOL, categoria datelor. În funcție de informațiile furnizate prin aceste clauze compilatorul COBOL alocă fiecărei date cantitatea de memorie necesară reprezentării valorilor sale. De exemplu, articolului ART-I, a cărui structură este:

```
01 ART-I.
05 Z1 PIC 9(3).
05 Z2 COMP-1.
05 Z3 PIC X(6).
05 Z4 COMP-2.
05 Z5 PIC 9(5) COMP-3.
05 Z6 PIC S9(18) COMP.
```

Îi va fi asociată o zonă-memorie de lungime 40 locații. Structura acestei zone este prezentată în figura XIV.1, unde  $a$  reprezintă adresa de început a zonei-memorie asociate articolului ART-I (care este întotdeauna o adresă multiplu de 8), iar porțiunile hașurate reprezintă locațiile neutilizate datorită alinierii datelor descrise cu opțiunile COMP, COMP-1 și COMP-2 — la multiplu de 2, 4 și, respectiv, 8.

Din cele prezentate rezultă că limbajul COBOL definește, în general, un model de organizare a memoriei pe caractere — în cazul datelor descrise prin opțiunile DISPLAY și COMP-3 ale clauzei USAGE — și de semicuvinte, cuvinte și dublu-cuvinte — în cazul datelor descrise cu opțiunile COMP, COMP-1 și, respectiv, COMP-2.

Prin intermediul clauzei SYNCHRONIZED, al cărei format general este:

```
{ SYNCHRONIZED } { LEFT }
{ SYNC           } { RIGHT }
```

acest model de organizare poate fi modificat astfel încât și datelor elementare descrise cu opțiunea DISPLAY și COMP-3 să li se aloc, funcție de lungime, un anumit număr de cuvinte-memorie. Lungimea zonelor asociate acestor date reprezintă deci, cel mai

Fig.  
XIV.1







### Observatii

În cazul formatului 1 lungimea datei *nume-dată* nu trebuie să depășească 80 de caractere (capacitatea unei cartele). Dacă este utilizată clauza FROM, mnemonicul (care este un identificator COBOL) trebuie asociat, în paragraful SPECIAL-NAMES, cuvîntului rezerva

**SYSIN.** De exemplu, în programul următor:

ENVIRONMENT DIVISION.

## ENVIRONMENTAL CONFIGURATION SECTION.

**SPECIAL-NAMES.**

SYNIN IS CITITOR-CARTELE.

DATA DIVISION.

WORKING-STORAGE SECTION.

01 TIP-PROGRAM PIC X.

PROCEDURE DIVISION.

START.

## ACCEPT TIP-PROGRAM

FROM CITITOR-CARTELE.

instrucțiunea ACCEPT descrie operația de transfer, în zona asociată datei TIP-PROGRAM, a unui caracter reprezentat în prima coloană a cartelei.

În cazul formatului 2 lungimea datei *nume-dată* nu trebuie să depășească 72 de caractere (capacitatea unui rind de la mașina de scris). Unitatea periferică implicată în transfer poate fi indicată prin cuvântul rezervat **CONSOLE** sau printr-un mnemonic ce este asociat acestui cuvânt în paragraful **SPECIAL-NAMES**. Execuția instrucțiunii ce urmează lui **ACCEPT** are loc numai după ce operatorul furnizează un număr de caractere egal cu lungimea datei *nume-dată*.

În cazul formatului 3, în funcție de opțiunea utilizată — DATE, DAY sau TIME — pentru înregistrarea datei sau a orei curente, trebuie să se rezerve o zonă-memorie de 6, 5 sau respectiv, 8 locații, ca în exemplul următor:

WORKING-STORAGE SECTION.

01 DATA-SI-ORA-CURENTĂ.

05 DATA-AALLZZ PIC 9(6).

05 DATA-ZZZAA PIC 9(5).

05 ORA-CURRENTĂ PIC 9(8).

PROCEDURE DIVISION.

**START**

ACCEPT DATA-AALLZZ FROM DATE

ACCEPT DATA-ZZZAA FROM DAY

ACCEPT ORA-CURRENTA FROM TIME.

b. Instrucțiunea **DISPLAY**. Prezintă trei formate:

Format general 1:

$$\text{DISPLAY} \left\{ \begin{array}{l} \text{nume-dată-1} \\ \text{literal-1} \end{array} \right\} \left[ , \left\{ \begin{array}{l} \text{nume-dată-2} \\ \text{literal-2} \end{array} \right\} \right] \dots [\text{UPON mnemonic}]$$

### Format general 2:

$$\text{DISPLAY} \left\{ \begin{array}{l} \text{nume-dată-1} \\ \text{literal-1} \end{array} \right\} \left[ , \left\{ \begin{array}{l} \text{nume-dată-2} \\ \text{literal-2} \end{array} \right\} \right] \dots \text{UPON} \left\{ \begin{array}{l} \text{CONSOLE} \\ \text{mnemonic} \end{array} \right\}$$

### Format general 3:

**DISPLAY** { *nume-dată-1* } [ , { *nume-dată-2* } ] ... **UPON** { **SYSPUNCH** }  
*literal-1* *literal-2* *mnemonic*



Instrucțiunea **DISPLAY**, corespunzător celor trei formate generale, descrie operațiile de transfer și înregistrare consecutivă a valorilor datelor *nume-dată-1* sau *literal-1*, *nume-dată-2* sau *literal-2* ...:

- pe suportul unității standard a sistemului — în cazul formatului 1;
- la mașina de scris — în cazul formatului 2;
- pe suportul unității standard de perforare a sistemului — în cazul formatului 3.

Prin execuția instrucțiunii, în funcție de formatul utilizat, se realizează:

- tipărirea unui rând la imprimantă;
- tipărirea unui rând la mașina de scris;
- perforarea unei cartele.

Mnemonicul specificat prin clauza **UPON** trebuie asociat, în paragraful **SPECIAL-NAMES**, cuvântului cheie **SYSOUT** — în cazul utilizării formatului 1, **CONSOLE** — în cazul utilizării formatului 2, **SYSPUNCH** — în cazul utilizării formatului 3.

#### Observație

În funcție de formatul utilizat — 1, 2 sau 3 — suma lungimilor datelor citate în instrucțiunea **DISPLAY**, nu trebuie să depășească 120, 72 și, respectiv, 80 caractere.

Literele citate în instrucțiunea **DISPLAY** pot fi numerice, nenumerică sau constante figurative (exceptând constanta figurativă **ALL**). Dacă în instrucțiune figurează o constantă figurativă, va fi transferat și înregistrat un singur caracter. De exemplu, considerînd următoarea secvență de program:

```
WORKING-STORAGE SECTION.  
01 DATE-EXEMPLU.  
05 DATA-1 PIC X(5) VALUE 'ABCDE'.  
05 DATA-2 PIC 9(5) VALUE 125.  
PROCEDURE DIVISION.  
START.  
    DISPLAY DATA-1 SPACE SPACE DATA-2 99.6  
    DISPLAY SPACE DATA-1" DATA-2  
    UPON SYSPUNCH.
```

prin execuția celor două instrucțiuni **DISPLAY** se va realiza:

- tipărirea la imprimantă a unui rând de forma:

ABCDE 0012599.6;

- perforarea unei cartele, valoarea datei **DATA-1** fiind reprezentată în coloanele 2—6, iar cea a datei **DATA-2**, în coloanele 10—14.

Dacă valorile datelor *nume-dată-1*, *nume-dată-2* — citate în instrucțiunea **DISPLAY** — sînt reprezentate în cod complementar, în zecimal-împachetat, în virgulă mobilă simplă sau dublă precizie transferul și înregistrarea pe suport sînt precedate de conversia acestora în zecimal-despachetat — în cazul datelor descrise cu opțiunea **COMP** sau **COMP-3** — și în virgulă mobilă format extern — în cazul datelor descrise cu opțiunea **COMP-1** sau **COMP-2**. Un număr în virgulă mobilă format extern este de forma:

$[\pm] \text{ mantisă } E[\pm] \text{ exponent}$

unde: *mantisa* reprezintă o succesiune de 8 cifre (dacă opțiunea este **COMP-1**) sau 17 cifre (dacă opțiunea este **COMP-2**), conținînd și marca zecimală, iar *exponentul* este format din două cifre.



PROBLEME

1. Considerînd descrierea, în secțiunea WORKING, a următorului articol:

```
01 ARTICOL.  
05 A      PIC S9(5) VALUE - 123.  
05 B      PIC X(5) VALUE '456'.  
05 C      COMP.  
10 D      PIC S9(5) VALUE +523.  
10 E      PIC S9(11) VALUE -523.  
05 F      PIC 9(7) VALUE 56 COMP-3.  
05 G      COMP-1 VALUE 64.  
05 H      COMP-2 VALUE - 100.  
05 I      PIC A(6) VALUE 'ABC'.  
05 J      PIC X(3) VALUE 'AB' SYNC RIGHT.  
05 K      PIC 9(5) VALUE 345 SYNC LEFT.  
05 L      PIC A(6) VALUE 'ABCDE' SYNC RIGHT.
```

să se indice structura și conținutul zonei-memorie care îi este rezervată de compilator.

2. Fie un pachet de cartele, care conține date referitoare la produsele livrate beneficiarilor unei întreprinderi. Pe fiecare cartelă sînt înregistrate valorile următoarelor date: codul beneficiarului (3n), codul produsului (5n), cantitatea livrată (6n) și valoarea (9n). Ultima cartelă conține, în afara datelor referitoare la beneficiar, caracterul S în coloana 80, marcînd sfîrșitul pachetului de cartele.

Să se scrie un program care să editeze, pentru beneficiarul cu codul x, un raport de forma celui prezentat în figura XIV.3. Raportul va conține codurile produselor livrate beneficiarului x și valoarea acestor produse. Cartelele vor fi citite prin instrucțiunea ACCEPT, iar codul beneficiarului x și data zilei vor fi furnizate de operator la mașina de scris.

46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73
				L	I	V	R	A	R	I	L	E		D	E		P	R	O	D	U	S	E				
			P	E	N	T	R	U		B	E	N	E	F	I	C	I	A	R	U	L	:		X	X	X	
				L	A		D	A	T	A	:		X	X	.	X	X	.	X	X							
	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
	*		C	O	D		P	R	O	D	U	S		*			V	A	L	O	A	R	E		*		
	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
	*					X	X	X	X	X			*		X	X	X	X	X	X	X	X	X	X	X	*	
	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
			T	O	T	A	L		V	A	L	O	A	R	E	:		X	X	X	X	X	X	X	X	X	X

Fig. XIV.3



## CAPITOLUL XV

### FIȘIERE PE CARTELE. VALIDAREA DATELOR

#### 1. FIȘIERE PE CARTELE

Aceste fișiere pot fi prelucrate numai în acces secvențial și pot fi create sau consultate. Un articol este înregistrat în întregime pe o singură cartelă, iar aceasta nu poate conține decât un singur articol. Caracterele ce compun articolul sînt înregistrate fiecare într-o coloană a cartelei, lungimea maximă a acestuia fiind de 80 de caractere. Indiferent de lungimea articolelor, compilatorul COBOL asociază fișierelor pe cartele o *zonă-articol* de lungime 80 locații, în care sînt înregistrate valorile datelor articolelor ce sînt citite sau a celor ce urmează a fi scrise.

În cazul fișierelor pe cartele care sînt consultate, formatul articolelor este considerat întotdeauna fix; prin citire caracterele, care compun un articol, sînt înregistrate în zona-articol în *cod EBCDIC*.

În cazul fișierelor pe cartele care sînt create, formatul articolelor poate fi fix sau variabil; prin scriere (perforare) caracterele care compun un articol sînt înregistrate pe cartelă în *cod Hollerith*.

Sfîrșitul unui fișier pe cartele care este consultat este indicat de o cartelă specială, numită *marcă de fișier*, ce conține în primele patru coloane caracterele: EOF.

În ceea ce privește descrierea fișierelor pe cartele, operațiile la care pot fi supuse, precum și exploatarea acestora se pot face precizările prezentate în continuare.

a. **Asignarea fișierelor pe cartele.** Identificatorul de exploatare (idex-ul) asociat unui fișier pe cartele (în fraza SELECT) căruia îi este afectată o unitate standard a sistemului (unitatea de intrare sau de perforare) trebuie să fie SYSIN — în cazul în care fișierul este consultat — sau SYSPUNCH — în cazul în care fișierul este creat. Pentru un fișier pe cartele al cărui idex este SYSIN, articolele care îl compun trebuie să urmeze, în fișierul de lucrări, cartelei de comandă prin care se cere execuția programului în care acest fișier este definit. Dacă unitatea periferică afectată unui fișier pe cartele nu este o unitate a sistemului, idex-ul trebuie să fie o literă urmată de caracterele CR sau CP, care indică tipul unității periferice: cititor de cartele și, respectiv, perforator de cartele. În acest caz, adresa simbolică a unității periferice trebuie indicată în cartela de comandă ASSIGN, în care trebuie să figureze același idex. Rezultă din cele prezentate că, în cazul fișierelor pe cartele, formatul general al frazei SELECT este:

$$\text{SELECT nume-fișier ASSIGN TO } \left\{ \begin{array}{l} \text{SYSIN} \\ \text{SYSPUNCH} \\ \text{idex } \left\{ \begin{array}{l} \text{CR} \\ \text{CP} \end{array} \right\} \end{array} \right\}$$

b. **Reprezentarea datelor pe cartele.** Deoarece prin citirea/scrierea unui articol nu se realizează decât conversia cod Hollerith — cod EBCDIC/cod



EBCDIC — cod Hollerith, la descrierea datelor ce îl compun trebuie să fie respectate următoarele reguli:

— în șabloanele clauzei PICTURE pot fi utilizate numai simbolurile 9, A, X, V și S;

— valorile datelor trebuie să fie reprezentate în cod EBCDIC.

c. **Fișiere cu mai multe tipuri de articole.** În general, datele ce urmează a fi prelucrate automat sînt transpuse de pe documentele primare (fișe, facturi, comenzi, bonuri de materiale etc.) pe cartele și prelucrate ca fișiere. Uneori însă, datele referitoare la un obiect (persoană, produs, material etc.) nu pot fi înregistrate pe o singură cartelă, ci pe mai multe cartele succesive. În astfel de situații apare o contradicție între noțiunea de înregistrare logică (articol) și înregistrare fizică. Anterior, s-a precizat faptul că în cazul fișierelor pe cartele o înregistrare fizică (cartelă) conține un singur articol. Acest lucru este adevărat din punctul de vedere al SGF-ului (care realizează efectiv operațiile de acces la înregistrările fișierelor) deoarece unui fișier îi este asociată o singură zonă-articol, în/din care sînt citite/scrise succesiv toate articolele acestuia, iar prin execuția unei instrucțiuni READ/WRITE se realizează citirea/scrierea unui singur articol. Utilizatorul poate însă considera că articolul este înregistrat pe mai multe cartele succesive.

Prelucrarea unui fișier de acest tip impune utilizatorului să identifice, prin program, articolele fișierului. De obicei, acest lucru se realizează asociind grupului de cartele succesive — ce corespund unui articol — indicator cu valori distincte pentru fiecare cartelă din grup. De asemenea, fiecare cartelă a grupului trebuie să conțină aceeași valoare a unei date aleasă drept identificator (cheie) al articolului (această dată poate fi, de exemplu, un cod, un număr matricol, un nume de persoană etc.). Identificarea articolelor se realizează testînd, după fiecare citire a unei cartele, valorile indicatorului și/sau al identificatorului articolului.

*Exemplu.* Se consideră un fișier pe cartele care conține date referitoare la elevii unei școli. Datele unui elev (NR-MATRICOL, NUME, ADRESA, SITUAȚIA-ȘCOLARĂ) sînt reprezentate pe două cartele succesive, ultimele 35 de coloane ale celei de a doua cartele nefiind utilizate (fig. XV.1). În vederea identificării articolelor, în fiecare din cele două cartele care, logic, reprezintă un articol, a fost inclusă data TIP, ale cărei valori sînt 1 și, respectiv, 2; de asemenea, data NR-MATRICOL, apare și în cea de a doua cartelă. Se cere ca prin consultarea acestui fișier să se tipărească la imprimantă, prin instrucțiunea DISPLAY, un raport care să conțină un rînd pentru fiecare elev. Macheta acestui rînd este prezentată în figura XV.2.

CARTELA -1	TIP	NR-MATRICOL	NUME	ADRESA
	9	9(4)	A(25)	X(50)
CARTELA-2	TIP	NR-MATRICOL	SITUAȚIA-ȘCOLARĂ	
	9	9(4)	X(40)	X(35)

Fig. XV.1

	NR-MATRICOL		NUME		SITUAȚIA-ȘCOLARĂ
X(5)	9(4)	X(5)	A(25)	X(5)	X(40)

Fig. XV.2



Fișierul poate fi descris în două moduri: considerînd că este alcătuit din două tipuri de articole; considerînd că este alcătuit dintr-un singur tip de articole.

● Primul mod de rezolvare este prezentat în programul XV.1. S-a considerat că pentru fiecare elev există în fișierul FCART două cartele cu aceeași valoare a datei NR-MATRICOL. Practic, în acest program, nu se face o identificare a articolelor, testînd valorile datei TIP și NR-MATRICOL, ci se gestionează numai citirile, presupunînd că prima cartelă citită conține numărul matricol, numele și adresa unui elev X, iar cea de a doua cartelă,

```

ID DIVISION.
PROGRAM-ID. PV1.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT FCART ASSIGN TO SYSIN.
DATA DIVISION.
FILE SECTION.
FD FCART RECORDING F LABEL FFLORD OMITTED.
01 CARTELA-1.
    05 TIP-1 PIC 9.
    05 NR-MATRICOL-1 PIC 9(4).
    05 NUME-1 PIC A(25).
    05 ADRESA-1 PIC X(50).
01 CARTELA-2.
    05 TIP-2 PIC 9.
    05 NR-MATRICOL-2 PIC 9(4).
    05 SITUATIA-SCOLARA-2 PIC X(40).
    05 FILLER PIC X(35).
WORKING-STORAGE SECTION.
01 SF PIC 9 VALUE ZERO.
01 W-NUME PIC A(25).
PROCEDURE DIVISION.
START.
    OPEN INPUT FCART.
    READ FCART AT END MOVE 1 TO SF.
    PERFORM CITESTE-SERIE UNTIL SF NOT = ZERO.
    CLOSE FCART.
    STOP RUN.
CITESTE-SERIE.
    MOVE NUME-1 TO W-NUME.
    READ FCART AT END MOVE 1 TO SF.
    DISPLAY / NR-MATRICOL-2
    / W-NUME
    / SITUATIA-SCOLARA-2
    READ FCART AT END MOVE 1 TO SF.
PROGRAMUL XV.1

```



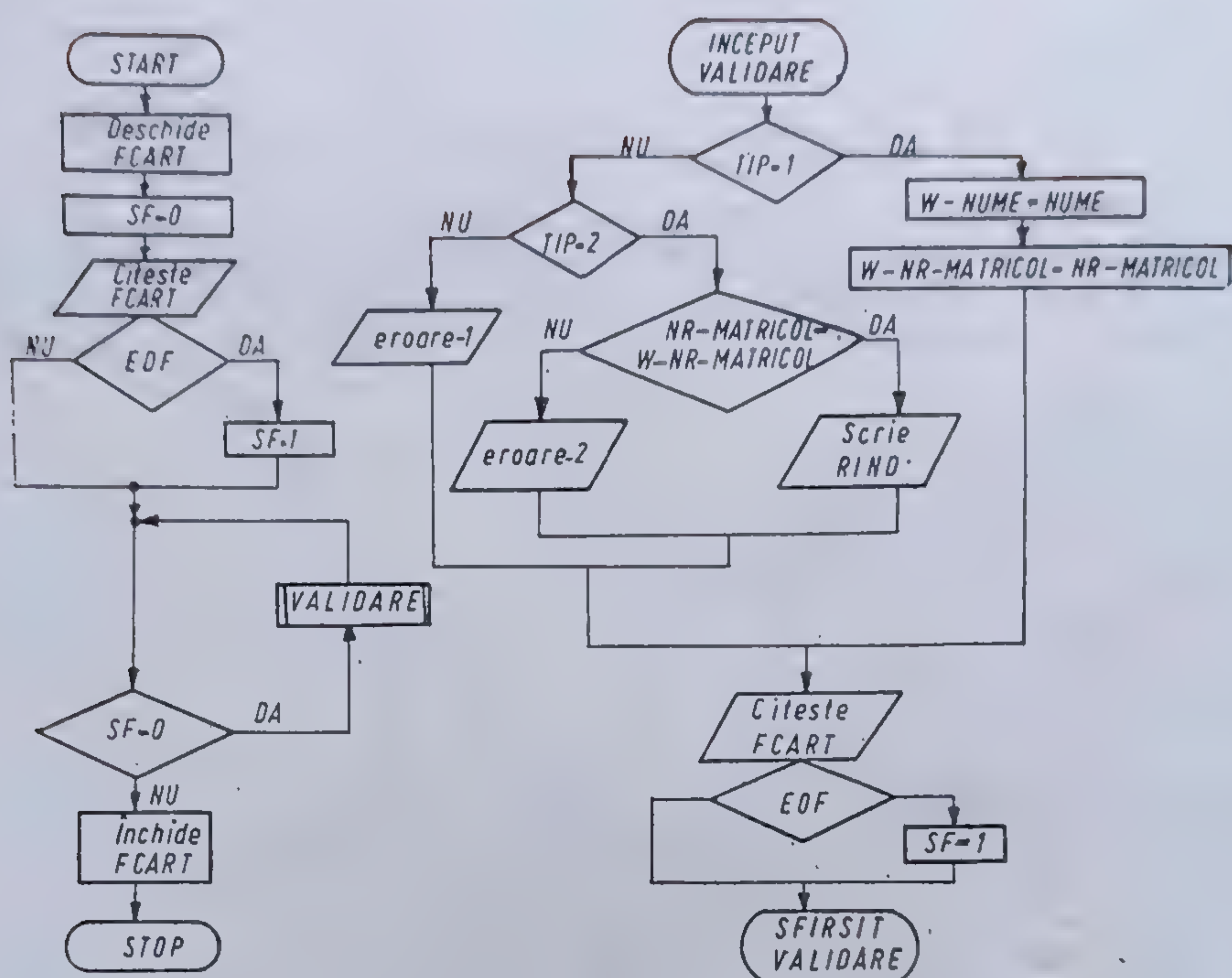


Fig. XV.3

situația școlară a aceluiași elev X. Pentru a se evita calificarea, articolelor și datelor care le compun le-au fost asociați identificatori distincți. De asemenea, întrucât fișierele pe cartele care sînt consultate trebuie să aibă articole de format fix, articolului CARTELA-2 i s-a asociat o dată (FILLER) de lungime 35 caractere, pentru a avea aceeași lungime ca articolul CARTELA-1.

Se mai poate face observația că, după citirea unei cartele de tip 1, valoarea datei NUME este atribuită datei W-NUME, pentru a fi păstrată în vederea tipăririi; acest lucru este necesar deoarece toate cartelele sînt citite în aceeași zonă-articol.

De asemenea, în instrucțiunea DISPLAY ar fi putut fi citat identificatorul NR-MATRICOL-1 în locul identificatorului NR-MATRICOL-2, deoarece ambii au asociată aceeași zonă în cadrul zonei-articol

● Cel de-al doilea mod de rezolvare este prezentat în programul XV.2. Algoritmul de prelucrare este mai complicat (fig. XV.3) în sensul că articolele sînt identificate (prin testarea valorilor datelor TIP și NR-MATRICOL), semnalîndu-se, prin mesaje la imprimantă, anumite cazuri posibile de eroare.

d. Redefinirea datelor. În programul XV.2 fișierului FCART i-a fost asociat un singur tip de articol ART, prin intermediul clauzei REDEFINES, al cărei format general este.

*număr-nivel nume-dată-1 REDEFINES nume-dată-2*

Această clauză permite ca aceeași zonă de memorie să fie alocată pentru două date elementare sau grupate (identificate prin *nume-dată-1* și *nume-dată-2*) diferite. Structura și categoria acestor date pot să difere, însă lungimile lor trebuie să coincidă. Facilitînd prelucrarea în aceeași zonă de memorie, în momente diferite, a unor date cu caracteristici diferite, clauza REDE-



```

ID DIVISION.
PROGRAM-ID. PV2.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT FCART ASSIGNED TO SYSIN.
DATA DIVISION.
FILE SECTION.
FD FCART RECORDING F LABEL RECORD OMITTED.
01 ART.
    05 TIP PIC 9.
    05 NR-MATRICOL PIC 9(4).
    05 CARTELA-1.
        10 NUME PIC A(25).
        10 ADRESA PIC X(50).
    05 CARTELA-2 REDEFINES CARTELA-1.
        10 SITUATIA-SCOLARA PIC X(40).
        10 FILLER PIC X(35).
WORKING-STORAGE SECTION
01 SF PIC 9 VALUE 0.
01 W-NUME PIC A(25).
01 W-NR-MATRICOL PIC 9(4).
PROCEDURE DIVISION.
START.
    OPEN INPUT FCART
    READ FCART AT END MOVE 1 TO SF
    PERFORM VALIDARE UNTIL SF = 1
    CLOSE FCART
    STOP RUN.
VALIDARE.
    IF TIP = 1
        THEN
            MOVE NUME TO W-NUME
            MOVE NR-MATRICOL TO W-NR-MATRICOL
        ELSE
            IF TIP = 2
                THEN
                    IF NR-MATRICOL = W-NR-MATRICOL
                        THEN
                            DISPLAY W-NUME W-NR-MATRICOL
                                SITUATIA-SCOLARA
                        ELSE
                            DISPLAY '*** NR-MATRICOL ERONAT ***'
                    ELSE
                        DISPLAY '*** TIP CARTELA ERONAT ***'
                ELSE
                    READ FCART AT END MOVE 1 TO SF.
PROGRAMUL XV.2

```

FINES permite reutilizarea dinamică a zonelor de memorie. Datele *nume-dată-1* și *nume-dată-2* trebuie să aibă același număr de nivel; de asemenea, rubrica în care este descrisă data *nume-dată-2* trebuie să urmeze celei în care este descrisă data *nume-dată-1*, iar între aceste două rubrici nu trebuie să apară decât, eventual, rubrici în care sînt descrise date subordonate datei *nume-dată-1*.

În cazul unui fișier care conține mai multe tipuri de articole, deoarece compilatorul îi asociază o singură zonă-articol, avem de-a face cu o alocare



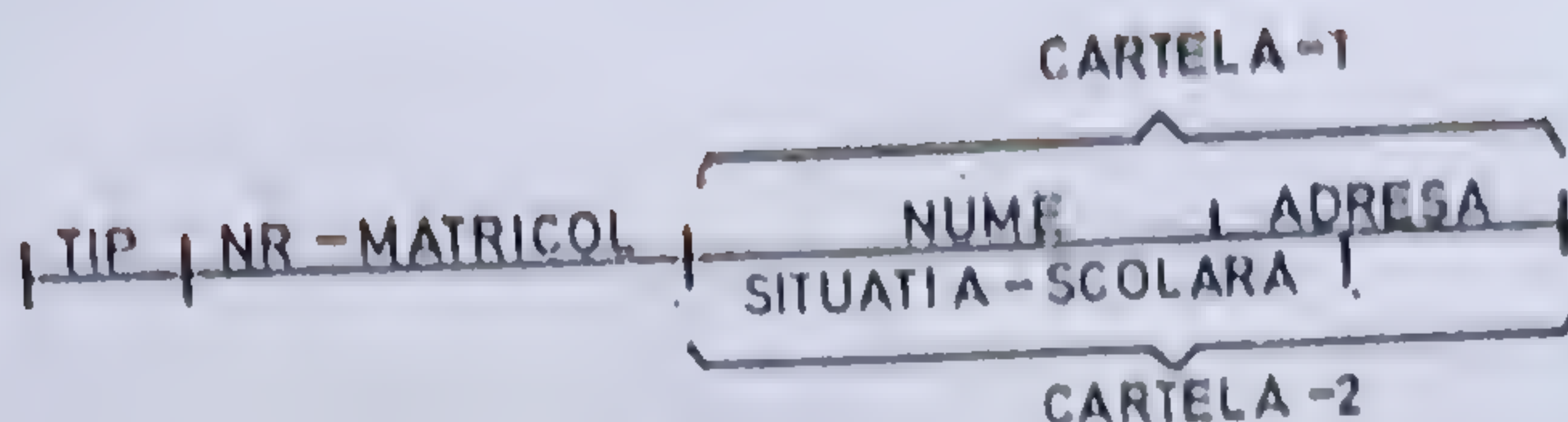


Fig. XV.4

implicită a acesteia, sau, altfel spus, o redefinire implicită a articolelor. Din această cauză, utilizarea clauzei REDEFINES în rubricile ce descriu articole (rubrici cu număr de nivel 01) ale fișierelor este interzisă. În schimb pot fi redefinite articolele descrise în secțiunea WORKING-STORAGE.

În cazul articolului ART al fișierului FCART din programul XV.2 redefinirea datei CARTELA-1 s-a făcut pentru a evita descrierea dublă a datelor TIP și NR-MATRICOL. Figura XV.4 indică sugestiv operația de redefinire: primele 5 locații ale zonei-articol sînt alocate datelor TIP și NR-MATRICOL, iar următoarele 75 locații, datelor CARTELA-1 și CARTELA-2, care au structuri diferite, dar aceeași lungime. Referirea datelor ce aparțin datelor CARTELA-1 și CARTELA-2 este funcție de valoarea datei TIP: dacă această valoare este 1, sînt referite datele subordonate datei CARTELA-1, iar dacă această valoare este 2, datele subordonate datei CARTELA-2 (fig. XV.4).

e. **Instrucțiunea EXAMINE.** În practică, nu întotdeauna, la reprezentarea pe cartele a valorilor datelor numerice, sînt perforate și zerourile nesemnificative. Astfel de situații apar, de regulă, atunci cînd numărul cifrelor semnificative ale valorilor datelor este mai mic decît numărul de coloane ale cîmpurilor în care acestea trebuie reprezentate (pentru operator fiind mai comod să lase coloane libere, decît să perforeze zerouri). Dacă aceste date participă ca operanzi în calcule este necesar ca, după citirea cartelei, în zonele-memorie în care au fost înregistrate valorile datelor, caracterul „spațiu” să fie înlocuit prin caracterul „zero”. Pentru a realiza această operație datele numerice trebuie redefinite, utilizînd clauza REDEFINES, ca date alfanumerice. Operația de înlocuire a spațiilor prin zerouri este descrisă prin instrucțiunea EXAMINE, în care sînt citate datele definite alfanumerice. În instrucțiunile aritmetice, în care datele apar ca operanzi, trebuie citat numele acelorași date, dar definite numerice.

### Exemplu

Fie un fișier pe cartele FCART, ale cărui articole ART conțin două date A și B, cărora nu li s-au reprezentat zerourile nesemnificative. După citirea fiecărui articol, valorile celor două date trebuie însumate, iar suma obținută se atribuie datei SUMA.

Descrierea articolului se poate realiza astfel:

```

01  ART.
02  A    PIC 9(5).
02  RA   REDEFINES A PIC X(5).
02  B    PIC 9(4).
02  RB   REDEFINES B PIC X(4).
  
```



Instrucțiunile care descriu operațiile enunțate sînt următoarele:

READ FCART

AT END MOVE 1 TO SF.

.

EXAMINE RA

REPLACING ALL SPACES BY ZEROS

EXAMINE RB

REPLACING ALL SPACES BY ZEROS

ADD A, B TO SUMA

Instrucțiunea EXAMINE descrie operația de numărare a aparițiilor unui caracter în cadrul valorii unei date și/sau înlocuirea acestui caracter printr-un alt caracter și are două formate generale:

Format general 1:

$$\text{EXAMINE } \textit{nume-dată-1} \text{ TALLYING } \left\{ \begin{array}{l} \text{ALL} \\ \text{LEADING} \\ \text{UNTIL FIRST} \end{array} \right\}$$
$$\left\{ \begin{array}{l} \textit{nume-dată-2} \\ \textit{literal-1} \end{array} \right\}$$
$$\left[ \text{REPLACING BY } \left\{ \begin{array}{l} \textit{nume-dată-3} \\ \textit{literal-2} \end{array} \right\} \right]$$

Format general 2:

$$\text{EXAMINE } \textit{nume-dată-1} \text{ REPLACING } \left\{ \begin{array}{l} \text{ALL} \\ \text{LEADING} \\ \text{[UNTIL] FIRST} \end{array} \right\}$$
$$\left\{ \begin{array}{l} \textit{nume-dată-2} \\ \textit{literal-1} \end{array} \right\} \text{ BY } \left\{ \begin{array}{l} \textit{nume-dată-3} \\ \textit{literal-2} \end{array} \right\}$$

unde:

— *nume-dată-1* reprezintă o dată numerică sau nenumerică, ale cărei valori sînt reprezentate în cod EBCDIC;

— *nume-dată-2* și *nume-dată-3* reprezintă date ale căror valori sînt formate dintr-un singur caracter;

— *literal-1* și *literal-2* reprezintă literale (nenumerică, numerice fără semn, sau constante figurative, cu excepția constantei ALL) formate dintr-un singur caracter.

.. Datele și literalele citate într-o instrucțiune EXAMINE trebuie să fie de aceeași categorie.

Prin execuția instrucțiunilor EXAMINE se realizează:

— numărarea aparițiilor, în valoarea datei *nume-dată-1*, a caracterului indicat prin *nume-dată-2* sau *literal-1* și, eventual, înlocuirea acestuia cu caracterul indicat prin *nume-dată-3* sau *literal-2* — în cazul formatului 1;

— înlocuirea, în valoarea datei *nume-dată-1*, a caracterului indicat prin *nume-dată-2* sau *literal-1*, cu caracterul indicat prin *nume-dată-3* sau *literal-2*

— în cazul formatului 2.



Semnificația opțiunilor ALL, LEADING, UNTIL FIRST și FIRST este următoarea;

- ALL, indică numărarea și/sau înlocuirea tuturor aparițiilor caracterului căutat (indicat prin *nume-dată-2* sau *literal-1*);
- LEADING, indică numărarea și/sau înlocuirea tuturor aparițiilor caracterului căutat, care sînt succesive și la începutul datei *nume-dată-1*;
- UNTIL FIRST, indică numărarea și/sau înlocuirea tuturor caracterelor care preced prima apariție a caracterului căutat;
- FIRST, indică înlocuirea primei apariții a caracterului căutat.

#### Observații

Numărarea și înlocuirea caracterului căutat au loc de la stînga la dreapta.

Dacă valorile datei *nume-dată-1* reprezintă numere cu semn, acesta este neglijat.

Rezultatul numărării apariției caracterului căutat este înregistrat într-un registru special, rezervat implicit de compilator la orice program COBOL. Acest registru poate fi referit în program prin cuvîntul rezervat TALLY. Deoarece în acest registru valorile sînt reprezentate în cod complementar semicuvînt, ele nu pot fi tipărite decît după ce sînt atribuite unei date ale cărei valori sînt reprezentate în cod EBCDIC.

În tabelul XV.1 sînt prezentate cîteva operații descrise prin instrucțiunea EXAMINE.

TABELUL XV.1

Instrucțiune EXAMINE	Valoarea datei ALFA		Conținutul zonei TALLY
	Înainte	După	
EXAMINE ALFA TALLYING UNTIL FIRST 'A'	AABAC	AABAC	0
EXAMINE ALFA TALLYING ALL 'A'	AABAC	AABAC	3
EXAMINE ALFA TALLYING LEADING 'A'	AABAC	AABAC	2
EXAMINE ALFA TALLYING UNTIL FIRST 'B' REPLACING BY 'X'	AABAC	XXBAC	2
EXAMINE ALFA TALLYING ALL 'A' REPLACING BY 'Y'	AABAC	YYBYC	3
EXAMINE ALFA TALLYING LEADING 'B' REPLACING BY 'Z'	AABAC	AABAC	0
EXAMINE ALFA REPLACING FIRST 'A' BY 'B'	AABAC	BABAC	—
EXAMINE ALFA REPLACING UNTIL FIRST 'D' BY 'W'	AABAC	WWWWW	—
EXAMINE ALFA REPLACING LEADING 'A' BY 'K'	AABAC	KKBAC	—
EXAMINE ALFA REPLACING ALL 'C' BY 'R'	AABAC	AABAR	—

f. Instrucțiuni pentru prelucrarea fișierelor pe cartele. Aceste instrucțiuni sînt cunoscute: OPEN, CLOSE, READ și WRITE. Formatele generale complete ale instrucțiunilor READ și WRITE sînt:

**READ** *nume-fișier* [INTO *nume-dată*] AT END *instrucțiune*.

**WRITE** *nume-articol* [FROM *nume-dată*]

Semnificația opțiunilor INTO și FROM este următoarea:

— articolul citit din fișier este transferat atît în zona-articol cît și în zona asociată datei *nume-dată*, fiind deci disponibil pentru prelucrare în două zone distincte;



Fișier

Memorie

Fișier

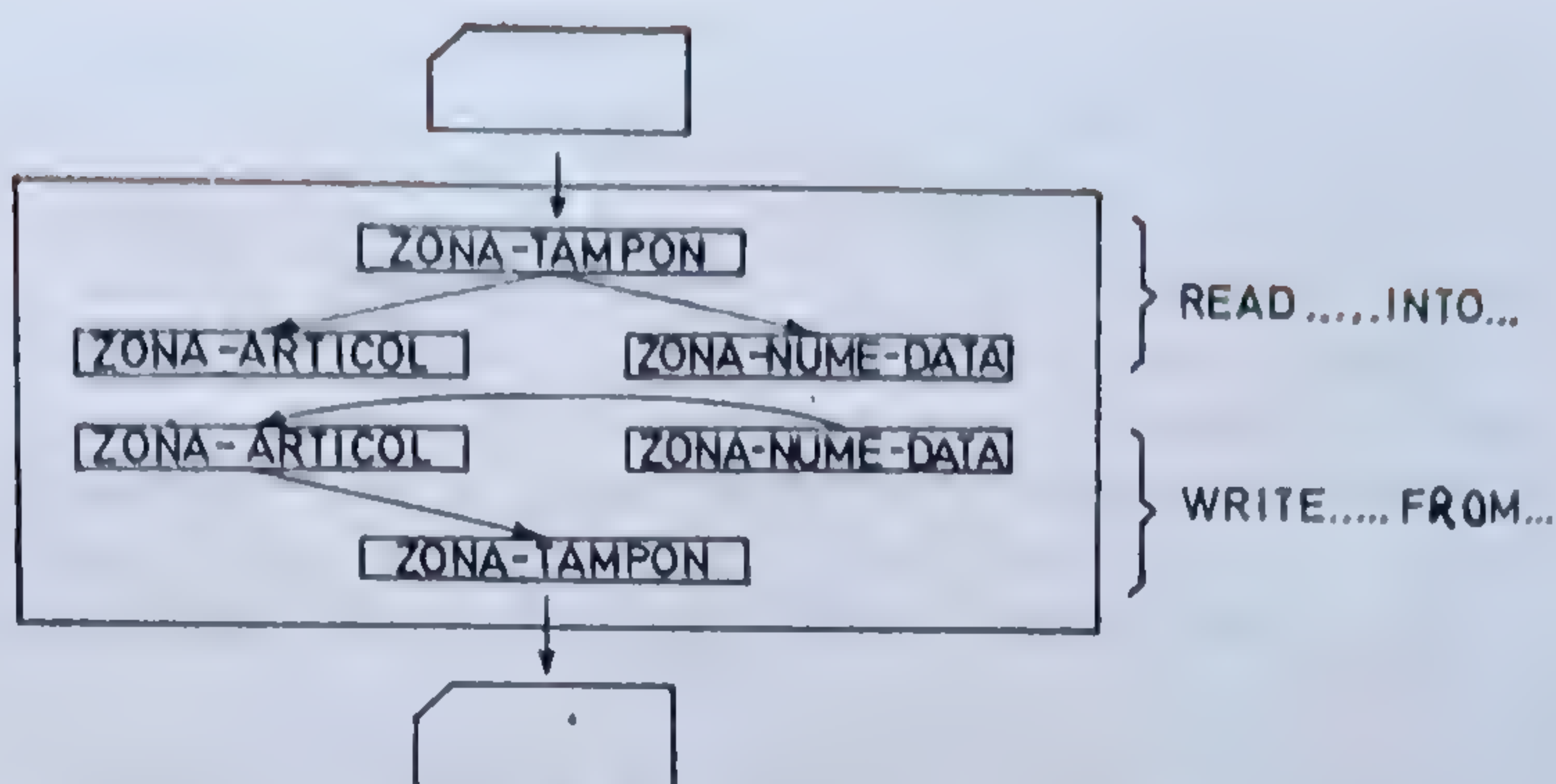


Fig. XV.5

— articolul care trebuie scris în fișier se află în zona asociată datei *nume-dată*, de unde este transferat în zona-articol.

Principiul de execuție al instrucțiunilor READ și WRITE, în care sînt citate opțiunile INTO și FROM este prezentat în figura XV.5. Aceeași figură sugerează faptul că transferul articolelor din/în fișier în/din zona-articol se realizează prin intermediul zonei-tampon.

## 2. TESTAREA CONDIȚIILOR

În limbajul COBOL testarea condițiilor se realizează prin instrucțiunile IF, PERFORM UNTIL, PERFORM VARYING și SEARCH. Există două categorii de condiții: *condiții simple* (numite și teste) și *condiții compuse*.

a. **Condiții simple.** Prin testarea unei condiții simple se poate determina:

- relația dintre doi termeni (*test de relație*);
- categoria (clasa) unei date (*test de clasă*);
- valoarea algebrică a unei date (*test de semn*);
- dacă valoarea unei date este egală cu valoarea altei date sau aparține unei mulțimi de valori (*test de nume de condiție*).

● **Testul de relație** stabilește dacă valoarea unui termen este mai mică, egală sau mai mare decît valoarea celuilalt termen. Termenii pot fi date, literale sau expresii aritmetice.

Formatul condiției testate este:

$$\left\{ \begin{array}{l} \text{nume-dată-1} \\ \text{literal-1} \\ \text{expresie-aritmetică-1} \end{array} \right\} \text{ IS [NOT] } \left\{ \begin{array}{l} \left\{ \begin{array}{l} \text{LESS THAN} \\ < \end{array} \right\} \\ \left\{ \begin{array}{l} \text{EQUAL TO} \\ = \end{array} \right\} \\ \left\{ \begin{array}{l} \text{GREATER THAN} \\ > \end{array} \right\} \end{array} \right\} \left\{ \begin{array}{l} \text{nume-dată-2} \\ \text{literal-2} \\ \text{expresie-aritmetică-2} \end{array} \right\}$$



Valoarea logică a condiției este „adevărat”, dacă cei doi termeni sînt în relația indicată de operatorul de relație și „fals”, în caz contrar. Dacă operatorul de relație este precedat de operatorul logic NOT, valoarea logică a condiției este „adevărat” dacă cei doi termeni nu sînt în relația indicată de operatorul de relație și „fals”, în caz contrar.

Utilizarea operatorului logic NOT suplinește astfel lipsa operatorilor de relație  $\leq$ ,  $\geq$  și  $\neq$ .

Compararea termenilor se realizează, în principiu, astfel:

— în cazul în care ambii termeni sînt numerici, se compară valorile algebrice ale acestora (valorile termenilor pot avea reprezentări interne diferite);

— în cazul în care cel puțin un termen este nenumeric, se compară succesiv, de la stînga la dreapta, caracterele ce compun valorile termenilor (dacă numărul caracterelor acestor valori nu este egal, compararea este precedată de completarea, la dreapta, cu caractere „spațiu”, a valorii termenului cu lungime mai mică; dacă unul dintre termeni este numeric, iar valoarea sa este reprezentată în zecimal-împachetat, în cod complementar sau în virgulă mobilă, compararea nu este precedată de conversia acestei valori în zecimal-despachetat).

● Testul de semn stabilește dacă valoarea algebrică a unei date elementare numerice sau a unei expresii aritmetice este mai mică, egală, sau mai mare decît zero.

Formatul condiției testate este:

$$\left\{ \begin{array}{l} \text{nume-dată} \\ \text{expresie-aritmetică} \end{array} \right\} \text{ IS [NOT] } \left\{ \begin{array}{l} \text{NEGATIVE} \\ \text{ZERO} \\ \text{POSITIVE} \end{array} \right\}$$

Valoarea logică a condiției este „adevărat” dacă valoarea datei sau a expresiei aritmetice corespunde opțiunii utilizate (NEGATIVE, ZERO sau POSITIVE) și „fals”, în caz contrar.

● Testul de clasă stabilește dacă valoarea unei date este numerică sau alfabetică.

Formatul condiției testate:

$$\text{nume-dată IS [NOT] } \left\{ \begin{array}{l} \text{NUMERIC} \\ \text{ALPHABETIC} \end{array} \right\}$$

Valoarea logică a condiției este „adevărat” dacă:

— valoarea datei *nume-dată* — care poate fi o dată elementară numerică reprezentată în zecimal-împachetat sau zecimal-despachetat, o dată alfa-numerică sau o dată grupată — este formată numai din cifre (cazul opțiunii NUMERIC);

— valoarea datei *nume-dată* — care poate fi o dată elementară nenumerică sau o dată grupată — este formată numai din litere, și, eventual caractere „spațiu” (cazul opțiunii ALPHABETIC).

● Testul numelui de condiție reprezintă o facilitate a limbajului COBOL care simplifică exprimarea unei condiții. Un nume de condiție reprezintă un



identificator asociat unei valori, unui interval de valori, sau unei reuniuni de intervale de valori ale unei date și este definit printr-o rubrică de forma:

88 nume-condiție  $\left\{ \begin{array}{l} \text{VALUE IS} \\ \text{VALUES ARE} \end{array} \right\} \text{literal-1} \left\{ \begin{array}{l} \text{THROUGH} \\ \text{THRU} \end{array} \right\} \text{literal-2}$   
 $\left[ \text{literal-3} \left\{ \begin{array}{l} \text{THROUGH} \\ \text{THRU} \end{array} \right\} \text{literal-4} \right] \dots$

Această rubrică trebuie să urmeze imediat celei în care este descrisă data a cărei valoare este testată. În cazul în care se testează dacă valoarea datei este egală cu o valoare dată, această valoare se indică prin *literal-1*. În cazul în care se testează dacă valoarea datei aparține unui interval de valori, limita inferioară a acestuia este indicată prin *literal-1*, iar limita superioară prin *literal-2*. În cazul în care se testează dacă valoarea datei aparține unei reuniuni de intervale, limitele acestora sînt indicate prin *literal-1* și *literal-2*, *literal-3* și *literal-4* ș.a.m.d.

Evident, literalele citate în rubrica ce descrie numele de condiție trebuie să corespundă categoriei datei ale cărei valori sînt testate.

În instrucțiunea care testează valoarea datei, condiția este indicată prin numele de condiție (care poate fi precedat de operatorul logic NOT). Valoarea logică a condiției este „adevărat” dacă valoarea datei corespunde valorii sau intervalelor de valori asociate numelui de condiție.

*Exemplu.* Se consideră că articolele unui fișier pe cartele conțin patru date: TIP, COD, DENUMIRE și VALOARE, care trebuie testate conform schemei din figura XV.6.

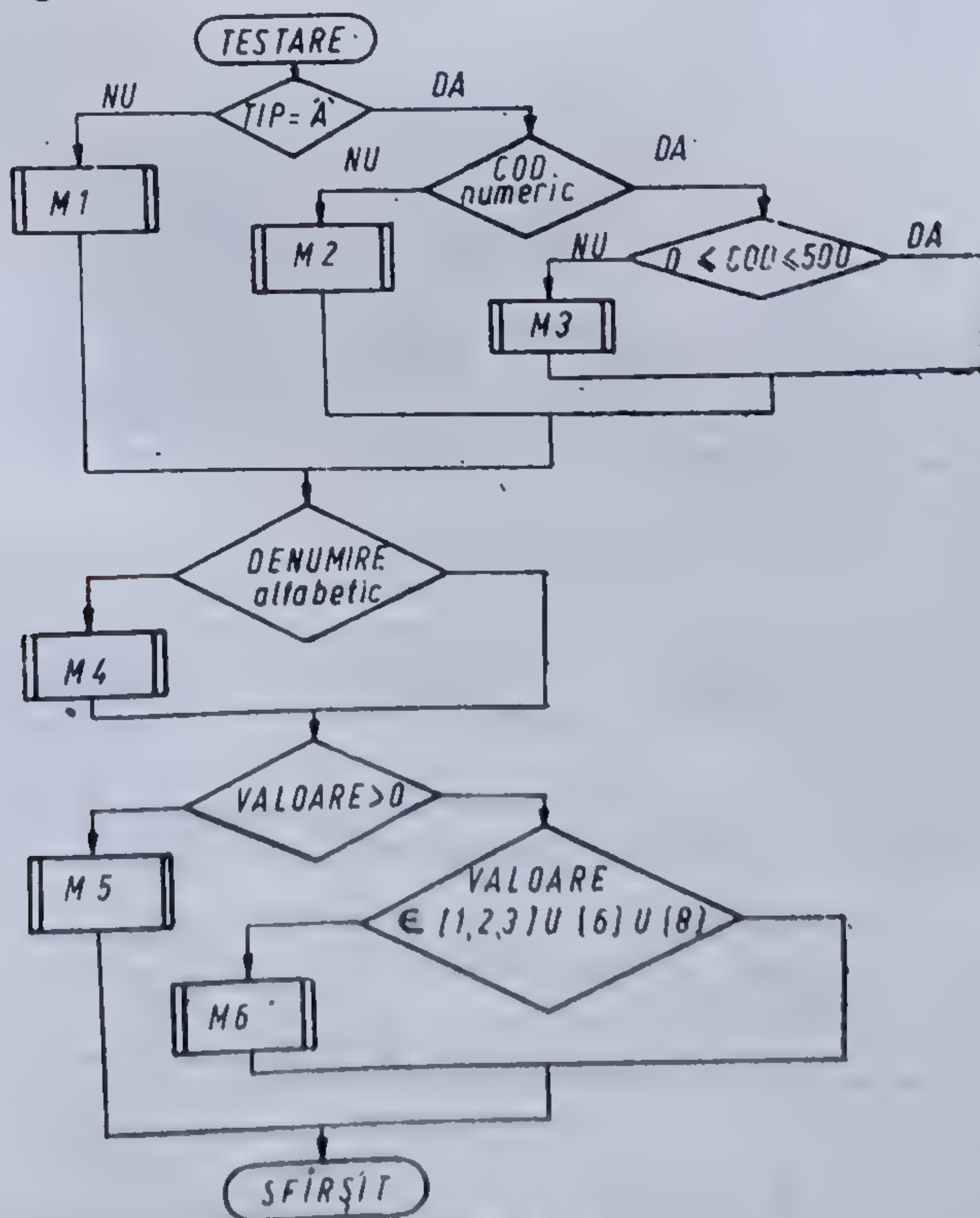


Fig. XV.6



```

01 ARTICOL.
  05 TIP          PIC X.
  88 TIP-CORECT VALUE 'A'.
  05 COD          PIC 9(5).
  88 COD-CORECT VALUE 0 THRU 500.
  05 DENUMIRE     PIC A(25).
  05 VALOARE      PIC 99(6).
  88 VAL-CORECTA VALUE 1 THRU 3, 6, 8.

```

```

TESTARE.
  IF NOT TIP-CORECT
  THEN
    PERFORM M1.
  ELSE
    IF COD NOT NUMERIC
    THEN
      PERFORM M2
    ELSE
      IF NOT COD-CORECT
      THEN
        PERFORM M3.
      IF DENUMIRE NOT ALPHABETIC
      THEN:
        PERFORM M4.
      IF VALOARE NOT POSITIVE
      THEN
        PERFORM M5
      ELSE
        IF NOT VAL-CORECTA
        THEN
          PERFORM M6.

```

Fig. XV.7

Descrierea articolelor și a testelor se poate realiza ca în figura XV.7.  
b. **Condiții compuse.** În alcătuirea condițiilor compuse pot intra condițiile simple, operatorii logici NOT, AND și OR și, eventual, parantezele.

Ca și în cazul condițiilor simple, valoarea logică a unei condiții compuse poate fi „adevărat” sau „fals”.

Dacă se consideră două condiții simple — C1 și C2 — care formează, prin intermediul operatorului logic AND și, respectiv, OR, o condiție compusă, valoarea logică a acesteia se stabilește conform tabelului XV.2. În această tabelă, prin *a* și *f* au fost notate valorile logice „adevărat” și, respectiv, „fals” ale condițiilor simple și ale celor compuse. Analizând această tabelă se poate afirma, prin extensie, că:

— valoarea logică a unei condiții compuse, formată din două sau mai multe condiții simple, legate prin operatorul logic OR, este „adevărat”

TABELUL XV.2

Condiții simple		Condiția compusă	
C1	C2	C1 AND C2	C1 OR C2
a	a	a	a
a	f	f	a
f	a	f	a
f	f	f	f



dacă valoarea logică a cel puțin uneia dintre condițiile simple este „adevărat”;

— valoarea logică a unei condiții compuse, formată din două sau mai multe condiții simple, legate prin operatorul logic AND, este „adevărat”, dacă valoarea logică a fiecăreia dintre condițiile simple este „adevărat”.

Utilizând condiții compuse secvența TESTARE din figura XV.6 se poate transcrie astfel:

```
TESTARE
  IF NOT TIP-CORECT
  THEN
    PERFORM M1
  ELSE
    IF COD NOT NUMERIC
    THEN
      PERFORM M2
    ELSE
      IF (COD < 0 AND COD > 500)
      THEN
        PERFORM M3.
  IF DENUMIRE NOT ALPHABETIC
  THEN
    PERFORM M4.
  IF VALOARE NOT POSITIVE
  THEN
    PERFORM M5
  ELSE
    IF NOT (VALOARE = 1 OR
            VALOARE = 2 OR
            VALOARE = 3 OR
            VALOARE = 6 OR
            VALOARE = 8)
    THEN
      PERFORM M6.
```

#### Observații

În absența parantezelor în cadrul unei condiții compuse, ierarhia operatorilor este următoarea:

- operatorii aritmetici;
- operatorii de relație;
- operatorul logic NOT;
- operatorul logic AND;
- operatorul logic OR.

De exemplu, considerând două date A și B, ale căror valori sînt 1 și, respectiv, -15, valoarea logică a condiției compuse

$$A + 5 > B \text{ AND } A < 5 \text{ OR } B < -20$$

este „adevărat”.

Prezența parantezelor în cadrul unei condiții compuse (care este, în general indicată) impune respectarea regulilor precizate la evaluarea expresiilor aritmetice.

De exemplu, considerînd aceleași valori pentru datele A și B, valoarea logică a condiției compuse:

$$\text{NOT } (((A < B) \text{ AND } (A > -5)) \text{ OR NOT } (B > 10))$$

este „fals”.



### 3. VALIDAREA DATELOR

Reprezintă operația prin care valorile datelor de intrare ale unui program (date organizate, în general, în fișiere) sînt verificate pentru a fi semnalate și înlăturate erorile care pot apărea datorită: întocmirii sau completării incorecte a documentelor primare de informații, transpunerii incorecte a informațiilor din documentele primare pe diferite suporturi de informație (cartele, benzi magnetice etc.), codificărilor greșite etc. Semnalarea și înlăturarea erorilor este necesară pentru a se evita ca programele de calcul să prelucrez date eronate.

Există două categorii de controale ce pot fi efectuate asupra valorii datelor: controale formale și controale de conținut (fond).

● *Controalele formale* au scopul de a verifica dacă:

- valorile datelor există în zonele ce le sînt asociate pe suport (de exemplu, este posibil ca valoarea unei date să nu fie perforată pe cartele);
- valorile datelor sînt decalate sau trunchiate;
- valorile datelor numerice sînt formate numai din caractere numerice, iar valorile datelor alfabetice sînt formate numai din caractere alfabetice (de exemplu, este posibil ca o literă să fie perforată în locul unei cifre sau invers).

● *Controalele de conținut* au ca scop de a verifica:

- compatibilitatea datelor, adică relațiile logice care există între valorile datelor ce compun articolele (de exemplu, în cazul unui articol ce conține date referitoare la un produs, se poate verifica relația: valoare = cantitate \* preț).
- dacă valorile datelor aparțin unor nomenclatoare sau norme de existență (de exemplu, în cazul unui articol ce conține date referitoare la un lucrător al unei întreprinderi, se poate verifica dacă numărul de copii are valori cuprinse între 1 și 18) etc.

În general, validarea datelor se realizează în cadrul unor programe distincte. Aceste programe trebuie realizate în așa fel, încît, la semnalarea unei erori, în cazul validării datelor unui articol, să nu fie abandonată efectuarea celorlalte controale, adică se va avea în vedere ca printr-o singură execuție a programelor să fie semnalate toate erorile posibile. De asemenea, la înlăturarea controalelor, trebuie să se aibă în vedere eliminarea efectuării de controale inutile (de exemplu, nu are sens verificarea relației valoare = cantitate \* preț, dacă valorile celor trei date nu sînt numerice).

În general, erorile sînt semnalate la imprimantă prin mesaje clare, iar aceste mesaje sînt precedate sau urmate de articolul la ale cărui date au fost detectate erorile. De asemenea, valorile datelor care nu satisfac condițiile testate sînt corectate, fiind supuse din nou operației de validare, printr-o nouă execuție a programului.

*Exemplu.* Se consideră un fișier pe cartele ale cărui articole conțin informații referitoare la produsele vîndute în cadrul magazinelor unei întreprinderi comerciale. Fiecare articol conține date referitoare la un produs: cod magazin, cod produs, denumire produs, cantitate, unitate de măsură, preț unitar și



COD - MAGAZIN	DENUMIRE - MAGAZIN	COD - PRODUS	DENUMIRE - PRODUS	UNITATE - MĂSURĂ	CANTITATE	PREȚ-UNITAR	VALOARE
9(2)	A(15)	9(3)	X(10)	A(3)	9(5)	9(4)V99	9(10)V99

Fig.  
XV.8

valoare (fig. XV.8). În vederea unor prelucrări ulterioare valorile datelor ce compun articolele trebuie validate pentru a fi depistate eventualele erori care au putut fi generate la întocmirea documentelor primare, transpunerea datelor din documentele primare pe cartele, perforarea cartelelor. Testele ce se vor face trebuie să verifice dacă:

— valorile datelor CANTITATE, PREȚ-UNITAR ȘI VALOARE sînt numerice;

— data COD-MAGAZIN are valori cuprinse între 1 și 10;

— relația  $VALOARE = CANTITATE * PREȚ-UNITAR$  este adevărată (testarea acestei relații se va face numai dacă valorile datelor VALOARE, CANTITATE și PREȚ-UNITAR sînt numerice);

— valorile datei UNITATE-MĂSURĂ aparțin mulțimii {KG, BUC, MP};

— valorile datei DENUMIRE-MAGAZIN există în câmpul care îi este rezervat pe cartelă și, eventual, dacă aceste valori sînt decalate spre dreapta.

În cazul în care este depistată o eroare aceasta va fi semnalată la imprimantă printr-un mesaj. De asemenea, dacă pentru o cartelă va fi semnalată cel puțin o eroare, la imprimantă va fi tipărită imaginea cartelei (aceasta trebuie însă să precedă mesajele de eroare). Se cere, de asemenea, să se determine și să se tipărească numărul total de cartele verificate, corecte și eronate.

O modalitate de rezolvare a problemei enunțate este prezentată în programul XV.3.

#### PROBLEME

1. Se consideră un fișier pe cartele ale cărui articole conțin date referitoare la furnizorii unei întreprinderi. Datele unui furnizor sînt înregistrate pe mai multe cartele astfel: datele de identificare a furnizorului — cod, denumire, adresa, distanța față de întreprinderea beneficiară — pe o cartelă (de tip 1); datele referitoare la materialele pe care furnizorul le livrează — cod, denumire, preț, costul transportului (exprimat în lei/ton/km) — sînt înregistrate pe mai multe cartele (de tip 2) ce urmează primei cartele.

Să se scrie un program care să determine și să afișeze la imprimantă datele de identificare a furnizorului care livrează cel mai convenabil un anumit material. Codul acestui material va fi citit de pe o cartelă parametru, prin instrucțiunea ACCEPT.

2. Zilnic, pentru fiecare muncitor al unei întreprinderi se perforază o cartelă cu datele de identificare a muncitorului (număr marcă, nume) și cite o cartelă pentru fiecare operație realizată (număr marcă, cod operație, număr de execuții a operației, valoarea manoperei pentru o execuție a operației) aceste cartele, ordonate după numărul de marcă, alcătuiesc un fișier.

Să se scrie un program care, pentru fiecare muncitor, să perforoze o cartelă care să conțină următoarele date: număr marcă, nume, retribuție.



## CAPITOLUL XVI

### ÎNTOCMIREA RAPOARTELOR

În capitolul XI s-a precizat faptul că limbajul COBOL prezintă, comparativ cu alte limbaje, facilități suplimentare pentru definirea și realizarea de *rapoarte* (situații tipărite la imprimantă). Scopul acestui capitol este de a prezenta aceste facilități precum și două operații implicate în realizarea rapoartelor: *transferul datelor* și *editarea datelor*. De asemenea, întrucât rapoartele sînt realizate, în general, prin intermediul fișierelor la imprimantă, vor fi prezentate și caracteristicile acestora.

#### 1. TRANSFERUL DATELOR

Operațiile de transfer (atribuire) sînt descrise în limbajul COBOL prin instrucțiunea MOVE.

Format general 1:

$$\text{MOVE} \left\{ \begin{array}{l} \text{nume-dată-1} \\ \text{literal-1} \end{array} \right\} \text{ TO } \text{nume-dată-2} [\text{,nume-dată-3}] \dots$$

Format general 2:

$$\text{MOVE} \left\{ \begin{array}{l} \text{CORRESPONDING} \\ \text{CORR} \end{array} \right\} \text{nume-dată-1 TO nume-dată-2}$$

Formatul 1 descrie operație de transfer a valorii datei *nume-dată-1*, *literal-1* (numită *dată emițătoare*) în zonele asociate datelor *nume-dată-2*, *nume-dată-3* ... (numite *date receptoare*).

Formatul 2 descrie operația de transfer a valorilor datelor corespondente ce aparțin datei *nume-dată-1* în zonele asociate datelor corespondente ce aparțin datei *nume-dată-2*.

#### Observație

Pentru simplificarea prezentării, în cele ce urmează se va considera că într-o operație de transfer este implicată o singură dată receptoare.

În funcție de tipul datelor citate în instrucțiunea MOVE transferul poate fi:

— *numeric*, dacă data receptoare este o dată elementară numerică sau numerică de editare;

— *alfanumeric*, dacă data receptoare este o dată elementară alfabetică, alfanumerică sau o dată grupată.



a. **Transferul numeric.** Se efectuează după următoarele reguli:

— caracterele valorii datei emițătoare sînt înregistrate în zona corespunzătoare datei receptoare de la marca zecimală spre stînga și, eventual, spre dreapta (dacă valoarea transferată reprezintă un număr cu zecimale);

— dacă numărul caracterelor valorii datei emițătoare este mai mic decît numărul locațiilor zonei asociate datei receptoare, în fiecare locație neocupată prin transfer este înregistrat caracterul 0 (zero); în caz contrar, caracterele ce nu pot fi înregistrate sînt neglijate (deci valoarea transferată este trunchiată);

— dacă formatul de reprezentare al valorilor datei receptoare este diferit de cel al valorilor datei emițătoare, transferul este precedat de conversia valorii de transferat în formatul de reprezentare corespunzător datei receptoare;

— dacă data receptoare este numerică de editare, prin transfer se realizează și editarea, conform șablonului de descriere din clauza PICTURE.

b. **Transferul alfanumeric.** Se efectuează după următoarele reguli:

— caracterele valorii datei emițătoare sînt înregistrate în zona corespunzătoare datei receptoare de la stînga la dreapta;

— dacă numărul caracterelor valorii datei emițătoare este mai mic decît numărul locațiilor zonei asociate datei receptoare, în fiecare locație neocupată prin transfer este înregistrat caracterul „spațiu”; în caz contrar, caracterele care nu pot fi înregistrate sînt neglijate;

— dacă valoarea de transferat este reprezentată în zecimal-împachetat, în cod complementar sau în virgulă mobilă, transferul este precedat de conversia valorii de transferat în zecimal-despachetat, în primele două cazuri, sau în virgulă mobilă format extern, în ultimul caz; un număr în virgulă mobilă format extern are forma:

$$\{\pm\} \text{ mantisă } E \{\pm\} \text{ exponent}$$

unde: *mantisă* este formată din 8 sau 17 cifre, după cum numărul este reprezentat în virgulă mobilă simplă precizie sau dublă precizie, iar *exponentul* este format din două cifre.

#### Observație

Regula de bază după care se efectuează transferul alfanumeric, în cazul în care data receptoare este o dată elementară alfanumerică sau alfabetică, poate fi modificată prin intermediul clauzei JUSTIFIED. Formatul general al acestei clauze este:

$$\left\{ \begin{array}{l} \text{JUSTIFIED} \\ \text{JUST} \end{array} \right\} \text{ RIGHT}$$

Clauza indică faptul că în zona asociată datei receptoare caracterele valorii transferate vor fi înregistrate de la dreapta la stînga, eventual, cu completare de spații, la stînga, sau trunchiere de caractere.

Exemplele următoare ilustrează principiul de execuție al instrucțiunii MOVE.

● Regulile de bază după care se efectuează transferul numeric și cel alfanumeric sînt relevate de exemplele din tabelul XVI.1.



TABELUL XVI.1

Data emițătoare		Data receptoare	
Șablon	Valoare	Șablon	Valoare
9(4)	123	9(5)	00123
9(4)	1234	9(3)	234
9(3)	123	9(3)V99	12300
9(3)V99:	123,45	9(4)	↑ 0123
9(3)V99	123,45	9(2)V9(3)	23450
A(4)	ABCD	A(6)	↑ ABCD␣
X(4)	1AB2	X(3)	1AB
A(4)	ABCD	A(6) JUST RIGHT	␣ABCD
X(4)	1AB2	X(3) JUST RIGHT	AB2
99V99	12,34	99.999	12.340
S9(4)	-1234	+9(6)	-001234
S9(3)V99	-123,45	-9(3).9	-123.4

● Se consideră următoarele două articole; primul este descris în secțiunea FILE, următorul, în secțiunea WORKING-STORAGE:

01 ARTICOL-CART.

05 COD	PIC 9(4).
05 DENUMIRE	PIC A(20).
05 CANTITATE	PIC 9(5).
05 PREȚ	PIC 9(3)V99.

01 ARTICOL-IMP.

05 FILLER	PIC X(10) VALUE SPACES.
05 DENUMIRE	PIC A (20).
05 FILLER	PIC X(5) VALUE SPACES.
05 COD	PIC 9(4).
05 FILLER	PIC X(5) VALUE SPACES.
05 CANTITATE	PIC 9(5).

Pentru a transfera valorile datelor COD, DENUMIRE și CANTITATE din primul articol, în zonele asociate acelorași date din cel de-al doilea articol, poate fi utilizată instrucțiunea:

MOVE CORRESPONDING ARTICOL-CART TO ARTICOL-IMP.

care este echivalentă cu următoarea secvență de instrucțiuni:

```

MOVE DENUMIRE OF ARTICOL-CART TO DENUMIRE OF
  ARTICOL-IMP
MOVE COD OF ARTICOL-CART TO COD OF ARTICOL-IMP
MOVE CANTITATE OF ARTICOL-CART TO CANTITATE OF
  ARTICOL-IMP

```

## 2. EDITAREA VALORILOR DATELOR

Editarea valorii unei date are loc în momentul înregistrării acesteia în zona care îi este asociată și constă în: inserarea semnului, a mărcii zecimale reale, înlocuirea zerourilor ne semnificative prin spații sau asteriscuri etc.



Aceste date se numesc *date numerice de editare*, iar valorile lor reprezintă, în general, numere pregătite pentru tipărire.

Simbolurile permise în șabloanele de descriere a datelor numerice de editare, precum și semnificația lor, sînt prezentate în tabelul XVI.2. Referitor la utilizarea lor se pot face observațiile prezentate în continuare.

TABELUL XVI.2

Simboluri permise în șabloanele de descriere

Simbolul	Semnificația
A	un caracter alfabetic
B	poziția în care va fi inserat caracterul „spațiu”
9	o cifră
V	marca zecimală virtuală
S	valorile datei reprezintă numere cu semn
0	poziția în care va fi inserat caracterul „zero”
Z	poziția în care un zero ne semnificativ va fi înlocuit prin caracterul „spațiu”
*	poziția în care un zero ne semnificativ va fi înlocuit prin caracterul *
.	marca zecimală reală, dacă aceasta este punctul, sau poziția în care va fi inserat caracterul ., dacă aceasta este virgula
,	marca zecimală reală, dacă aceasta este virgula, sau poziția în care va fi inserat caracterul ,, dacă aceasta este punctul
+	poziția în care va fi inserat caracterul +, dacă valoarea datei este un număr pozitiv, sau caracterul -, dacă valoarea datei este un număr negativ
-	poziția în care va fi inserat caracterul „spațiu” dacă valoarea datei este un număr pozitiv, sau caracterul -, dacă valoarea datei este un număr negativ
CR/DB	pozițiile în care vor fi inserate două caractere „spațiu”, dacă valoarea datei este un număr pozitiv, sau caracterele CR/DB, dacă valoarea datei este un număr negativ
\$ (sau alte simboluri)	poziția în care va fi inserat simbolul monetar

1. În mod obișnuit, marca zecimală reală este indicată prin caracterul, iar simbolul monetar prin caracterul \$. Schimbarea acestor convenții se poate realiza prin clauzele **CURRENCY SIGN IS literal** și **DECIMAL-POINT IS COMMA** ale paragrafului SPECIAL-NAMES. Prima clauză stabilește un alt simbol monetar (care nu poate fi însă unul dintre următoarele caractere: cifrele de la 0 la 9, literele A, B, C, D, E, P, R, S, V, X, Z sau caracterele speciale ., +, -, ,, (.), ', și spațiul), iar a doua clauză stabilește drept marcă zecimală reală virgula. Alegerea virgulei drept marcă zecimală necesită ca în literalele numerice această convenție să fie respectată.

În absența clauzei DECIMAL-POINT IS COMMA marca zecimală reală este punctul, deci acesta poate figura în șablon o singură dată, iar virgula poate fi utilizată ca separator al grupelor de cîte trei cifre; în caz contrar, funcțiile punctului și ale virgulei se schimbă între ele.

2. Editarea valorilor datelor se realizează prin două metode:  
 -- inserare (fixă sau mobilă);  
 -- suprimare și înlocuire.

● În cazul *inserării fixe*, caracterele de inserat sînt înregistrate în zona asociată datei, în locațiile ale căror poziții în cadrul zonei coincid cu poziția simbolurilor în cadrul șablonului. Simbolurile care indică o inserare



fixă sînt B, O,, (sau ,), +, —, CR, DB și \$ (sau un alt simbol monetar). Simbolurile +, — și simbolul monetar indică o inserare fixă, dacă apar o singură dată în șablon. De asemenea, trebuie remarcat faptul că simbolurile + și — pot figura (ori unul, ori celălalt) ca prime sau ultime simboluri ale șablonului, simbolul monetar, ca prim simbol al șablonului, iar simbolurile CR sau DB, ca ultime simboluri ale șablonului. Dacă simbolurile +, — și simbolul monetar apar de mai multe ori consecutiv în șablon (și anume, la începutul acestuia), ele indică o inserare mobilă și se numesc simboluri mobile).

• În cazul *inserării mobile*, caracterul de inserat este înregistrat, în zona asociată datei, în locația ce corespunde:

— primului simbol mobil al șablonului, dacă valoarea datei nu conține zerouri nesemnificative;

— ultimului simbol mobil al șablonului, dacă valoarea datei conține un număr de zerouri nesemnificative mai mare decît numărul de apariții al simbolului mobil în șablon.

— ultimului zero nesemnificativ al valorii datei, dacă aceasta conține un număr de zerouri nesemnificative mai mic decît numărul de apariții ale simbolului mobil în șablon.

La utilizarea simbolurilor mobile trebuie să se țină seama de următoarele reguli:

— dacă șablonul indică inserarea mărcii zecimale și simbolul mobil apare la dreapta acesteia, iar partea întreagă a valorii datei este zero și primele cifre ale părții zecimale sînt zerouri, simbolul mobil este înregistrat în locația ce o precede pe cea în care este înregistrată marca zecimală;

— dacă șablonul conține numai simboluri mobile, iar valoarea datei este zero, în fiecare locație a zonei asociată datei este înregistrat caracterul „spațiu” (chiar dacă șablonul indică inserarea mărcii zecimale);

— într-un șablon nu poate figura decît un singur simbol mobil.

• *Suprimarea și înlocuirea* se indică prin simbolurile mobile Z și \*. Aici trebuie făcută observația că suprimarea zerourilor nesemnificative și înlocuirea lor prin spații, sînt oprite de marca zecimală. Dacă însă valoarea datei este zero, iar șablonul este format numai din simboluri Z, sau numai din simboluri \*, în fiecare locație a zonei asociată datei este înregistrat caracterul „spațiu” sau, respectiv, caracterul \*.

În tabelul XVI.3 sînt prezentate cîteva exemple de editare a valorilor datelor.

TABELUL XVI.3

Valoarea datei	Șablon	Marcă zecimală/ simbol monetar	Conținutul zonei						
1234	99BB99	punct	<table><tr><td>1</td><td>2</td><td></td><td></td><td>3</td><td>4</td></tr></table>	1	2			3	4
1	2				3	4			
1234	990099		<table><tr><td>1</td><td>2</td><td>0</td><td>0</td><td>3</td><td>4</td></tr></table>	1	2	0	0	3	4
1	2		0	0	3	4			
12,34	99.99		<table><tr><td>1</td><td>2</td><td>.</td><td>3</td><td>4</td></tr></table>	1	2	.	3	4	
1	2	.	3	4					
1	ZZ99	<table><tr><td></td><td></td><td>0</td><td>1</td></tr></table>			0	1			
		0	1						
1	Z(4)	<table><tr><td></td><td></td><td></td><td>1</td></tr></table>				1			
			1						
0	Z(4)	punct	<table><tr><td></td><td></td><td></td><td></td></tr></table>						
0,01	ZZ.ZZ	<table><tr><td></td><td></td><td>.</td><td>0</td><td>1</td></tr></table>			.	0	1		
		.	0	1					



TABELUL XVI.3 (continuare)

Valoarea datei	Șablon	Marcă zecimală/ simbol monetar	Conținutul zonei									
0	ZZ.ZZ	punct	<table><tr><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>									
- 12	+\$9(4)	dolar	<table><tr><td>-</td><td>\$</td><td>0</td><td>0</td><td>1</td><td>2</td></tr></table>	-	\$	0	0	1	2			
-	\$	0	0	1	2							
+ 12	+++9		<table><tr><td></td><td>+</td><td>1</td><td>2</td></tr></table>		+	1	2					
	+	1	2									
- 1	--9		<table><tr><td></td><td>-</td><td>0</td><td>1</td></tr></table>		-	0	1					
	-	0	1									
0,01	\$\$.\$	dolar, punct	<table><tr><td></td><td>\$</td><td>.</td><td>0</td><td>1</td></tr></table>		\$	.	0	1				
	\$	.	0	1								
0	\$\$.\$	dolar, punct	<table><tr><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>									
12	\$+++9.99	dolar, punct	<table><tr><td>\$</td><td></td><td>+</td><td>1</td><td>2</td><td>.</td><td>0</td><td>0</td></tr></table>	\$		+	1	2	.	0	0	
\$		+	1	2	.	0	0					
12,34	**99		<table><tr><td>*</td><td>*</td><td>1</td><td>2</td></tr></table>	*	*	1	2					
*	*	1	2									
12 345,67	ZZ,Z99.99	punct	<table><tr><td>1</td><td>2</td><td>,</td><td>3</td><td>4</td><td>5</td><td>.</td><td>6</td><td>7</td></tr></table>	1	2	,	3	4	5	.	6	7
1	2	,	3	4	5	.	6	7				
4 567,890	9.999,999	virgulă	<table><tr><td>4</td><td>.</td><td>5</td><td>6</td><td>7</td><td>.</td><td>8</td><td>9</td><td>0</td></tr></table>	4	.	5	6	7	.	8	9	0
4	.	5	6	7	.	8	9	0				
12345	L9(5)	caracterul L	<table><tr><td>L</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr></table>	L	1	2	3	4	5			
L	1	2	3	4	5							
- 123456	9(6) -		<table><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>-</td></tr></table>	1	2	3	4	5	6	-		
1	2	3	4	5	6	-						
+ 1234	9(4) +		<table><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>+</td></tr></table>	1	2	3	4	+				
1	2	3	4	+								
12007	*(7)9		<table><tr><td>*</td><td>*</td><td>*</td><td>1</td><td>2</td><td>0</td><td>0</td><td>7</td></tr></table>	*	*	*	1	2	0	0	7	
*	*	*	1	2	0	0	7					
123	9(3)CR		<table><tr><td>1</td><td>2</td><td>3</td><td></td><td></td></tr></table>	1	2	3						
1	2	3										
456	9(3)DB		<table><tr><td>4</td><td>5</td><td>6</td><td></td><td></td></tr></table>	4	5	6						
4	5	6										
- 123	9(3)CR		<table><tr><td>1</td><td>2</td><td>3</td><td>C</td><td>R</td></tr></table>	1	2	3	C	R				
1	2	3	C	R								
- 456	9(3)DB		<table><tr><td>4</td><td>5</td><td>6</td><td>D</td><td>B</td></tr></table>	4	5	6	D	B				
4	5	6	D	B								

### 3. FIȘIERE LA IMPRIMANTĂ

a. **Caracteristici.** Fișierele la imprimantă pot fi numai create, iar modul de acces la articole care le compun este întotdeauna secvențial. Un articol reprezintă un rând tipărit pe hîrtia de imprimat. Numărul maxim de caractere care intră în alcătuirea unui rând este în funcție de tipul imprimantei. În cazul imprimanțelor compatibile cu sistemele FELIX acest număr este 132. Formatul articolelor poate fi fix sau variabil.

Imprimanta poate realiza trei tipuri de operații: imprimarea unui rând; avansul hîrtiei cu un anumit număr de rînduri și imprimarea unui rînd; avansul hîrtiei fără imprimare. Aceste operații, a căror efectuare este indicată în program, dă posibilitatea dispunerii rîndurilor pe hîrtia de imprimat în diverse moduri. În vederea programării acestor operații, articolelor li se asociază un caracter, numit *cod de salt*, a cărui valoare definește condițiile de imprimare (tabelul XVI.4).



TABELUL XVI.4

Cod de salt (hexazecimal)	Semnificație
EO	Avans interzis înainte și după imprimare
CO	Avans interzis înainte de imprimare
C1	Avans 1 rând înainte de imprimare
C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF	Avans $\left\{ \begin{matrix} 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \end{matrix} \right\}$ rânduri înainte de imprimare
F0 F1 F2 F3 F4 F5 F6 F7	Avans definit de canalul 0 al benzii pilot Avans definit de canalul $\left\{ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{matrix} \right\}$ al benzii pilot.

Observații

Codul de salt (care este primul caracter al articolelor fișierelor la imprimantă) nu este tipărit pe hîrtia de imprimat, dar trebuie să apară în descrierea articolelor. Aceasta înseamnă că lungimea maximă a articolelor este de 133 caractere.

Identificatorul de exploatare asociat unui fișier la imprimantă poate fi SYSOUT sau o literă urmată de caracterele PR.

b. Instrucțiuni pentru prelucrarea fișierelor la imprimantă. Aceste instrucțiuni sînt OPEN, CLOSE și WRITE. Instrucțiunea WRITE are două formate generale:

Format general 1:

WRITE *nume-articol* [FROM *nume-dată*]

Format general 2:

WRITE *nume-articol* [FROM *nume-dată-1*]

$\left\{ \begin{matrix} \text{BEFORE} \\ \text{AFTER} \end{matrix} \right\}$  ADVANCING  $\left\{ \begin{matrix} \textit{nume-dată-2} \text{ LINES} \\ \text{întreg LINES} \end{matrix} \right\}$

• Dacă este utilizat primul format, codului de salt trebuie să i se atribuie una din valorile permise (tabelul XVI.4).



*Exemplu:*

ENVIRONMENT DIVISION.  
CONFIGURATION SECTION.  
INPUT-OUTPUT SECTION.  
FILE-CONTROL.

SELECT FISIER-IMPRIMANTA ASSIGN TO SYSOUT.

DATA DIVISION.  
FILE SECTION.

FD FISIER-IMPRIMANTA RECORDING F  
LABEL RECORD OMITTED.

01 RIND.  
05 COS-SALT PIC X.  
05 GRUP PIC X(50).

PROCEDURE DIVISION.

MOVE 'C' TO COD-SALT  
MOVE ALL '\*' TO GRUP  
WRITE RIND

Prin execuția instrucțiunii WRITE RIND la imprimantă va fi tipărit un rând (ce conține 50 de caractere \*), însă după avansarea hîrtiei cu trei rînduri (valoarea codului de salt fiind C3<sub>16</sub>).

● În cazul formatului 2, clauza ADVANCING precizează condițiile în care va avea loc imprimarea. După cum este utilizată opțiunea BEFORE sau AFTER, imprimarea rîndului va avea loc înainte sau după avansarea hîrtiei cu un număr de rînduri. Acest număr depinde de valoarea datei *nume-dată-2* sau valoarea lui *întreg*.

Data *nume-dată-2* poate fi o dată numerică descrisă cu șablonul 9 sau 99, sau o dată alfanumerică, descrisă cu șablonul X. În primul caz valorile datei pot fi cuprinse între 0 și 99, semnificația acestora fiind:

0 — hîrtia nu avansează (supraimprimare);

1 — hîrtia avansează cu un rînd;

99 — hîrtia avansează cu 99 rînduri.

În cel de-al doilea caz, valorile datei și semnificația lor sînt:

*spațiu* — hîrtia avansează cu un rînd;

0 — hîrtia avansează cu două rînduri;

— — hîrtia avansează cu trei rînduri;

— — hîrtia nu avansează (supraimprimare);

+ — hîrtia avansează pînă la începutul unei noi pagini;

1







Tuturor datelor care compun articolele R1, R2, R3 și R4 li s-au atribuit valori inițiale. În schimb, în cazul articolului RÎND, numai anumitor date (cele identificate prin FILLER) li s-au atribuit valori (valori ce delimitează coloanele raportului); datelor COD-MAGAZIE-E, COD-MATERIAL-E și STOC-EXISTENT-E nu li s-au atribuit valori inițiale, deoarece valorile lor diferă de la un rând la altul. Pentru scrierea unui anumit rând, imaginea acestuia, aflată în zona asociată articolului R1, R2, R3, R4 sau RÎND, a fost transferată, utilizând opțiunea FROM a instrucțiunii WRITE, în zona-articol ART-E.

## PROBLEME

1. Considerînd următoarele descrieri de date:

01 DATE-EMITATOARE.

05 A PIC 9(4) VALUE 12.

05 B.

10 C PIC S9(4) VALUE -123.

10 D PIC X(4) VALUE '1234'.

05 E PIC 9(4)V99 VALUE 12.34.

10 DATE-RECEPTOARE:

05 G PIC S9(4)V99.

05 H PIC S9(3) COMP-3.

05 I PIC X(6) JUST RIGHT.

05 J PIC 9V9.

05 K PIC X(9)

să se indice valorile datelor receptoare după execuția următoarelor instrucțiuni:

MOVE A TO G

MOVE C TO H

MOVE D TO I

MOVE E TO J

MOVE B TO K

MOVE DATE-EMITATOARE TO DATE-RECEPTOARE.

2. Să se scrie un program care să citească  $n$  numere reale ( $n \leq 20$ ) de pe o cartelă și să le afișeze fiecare pe un rând (punîndu-li-se în evidență semnul și marca zecimală) astfel: primul număr va fi afișat la începutul unei pagini, iar celelalte, să fie precedate fiecare de un număr de interlinii egal cu numărul său de ordine.
3. Să se editeze raportul prezentat în figura XIII.5 (vezi programul XIII.1) astfel încît titlul acestuia să apară la începutul unei pagini, iar pe o pagină să fie scrise 60 de rînduri (în acest număr incluzîndu-se și numărul rîndurilor care formează antetul raportului).



## CAPITOLUL XVII

### PRELUCRAREA TABLOURILOR

În COBOL un *tablou* reprezintă o mulțime de date de aceeași categorie și lungime, care apar succesiv în cadrul unui articol; aceste date se numesc *elemente* ale tabloului. Există două categorii de tablouri: *cu număr fix de elemente* și *cu număr variabil de elemente*. În figura XVII.1 sînt prezentate patru tablouri. Primele trei au număr fix de elemente; elementele NOTA, A și LUNA reprezintă: notele anuale ale unui elev, elementele unei matrici cu patru linii și patru coloane — A (4,4)-și, respectiv, producțiile lunare obținute de o întreprindere industrială. Tabloul inclus în articolul ART, are un număr variabil de elemente; un element reprezintă o dată grupată, REPER care indică, prin valorile datelor subordonate, numărul de operații efectuate la un reper de către un lucrător al unei întreprinderi. Întrucît numărul de repere poate diferi de la un lucrător la altul, pentru prelucrarea elementelor tabloului, în articolul ART, s-a prevăzut data NR-REPERE ale cărei valori indică numărul de repere.

#### 1. IDENTIFICAREA ELEMENTELOR TABLOURILOR

Individualizarea elementelor unui tablou se poate realiza utilizînd *indici* sau *indecși*.

● Un *indice* reprezintă un literal numeric, o dată numerică sau registrul special TALLY. Valoarea unui indice trebuie să fie un număr întreg, mai mare decît zero. Numărul de indici (sau indecși) folosiți pentru identificarea unui element al unui tablou reprezintă *dimensiunea tabloului*. În COBOL pot fi prelucrate tablouri cu una, două și cu trei dimensiuni. În funcție de numărul de dimensiuni ale tabloului și de elementele care trebuie referite, identificarea se realizează utilizînd unul, doi sau trei indici (sau indecși). Un indice, sau un ansamblu de indici, se scrie între paranteze, la cel puțin un spațiu după numele elementului; dacă identificarea se realizează printr-un ansamblu de indici, aceștia trebuie separați prin cel puțin un spațiu sau prin virgulă urmată de cel puțin un spațiu (*nici un spațiu nu trebuie să urmeze parantezei deschise sau să preceadă paranteza închisă*). De exemplu, construcțiile:

NOTA (i)  
A (2, 1)  
LUNA (i, J, 1)  
NR-OPERATII (2)

identifică:

- a doua notă din tabloul TABLOU-NOTE, dacă  $I = 2$ ;
- primul element al liniei 2 a matricii A (4,4) din tabloul TABLOU-MATRICE;







— producția obținută în a doua lună a trimestrului 1 din semestrul 2, din tabloul TABLOU-PRODUCTIE, dacă  $I = 1$  iar  $J = 2$ ;

— numărul de operații efectuate pentru al doilea reper din tabloul TABLOU-REPERE.

#### Observații

Valoarea unui indice, utilizat pentru referirea unui element al unei dimensiuni a tabloului, trebuie să fie cuprinsă între 1 și numărul maxim de elemente ce corespund dimensiunii respective.

— Un indice nu poate fi indicat; de exemplu, construcția:

LUNA (I, J, K (L))

nu este admisă.

● Un *index* reprezintă un tip aparte de indice; spre deosebire de valoarea unui indice, valoarea internă a unui index reprezintă adresa relativă în cadrul tabloului, a zonei rezervată unui element. De exemplu, în cazul tabloului TABLOU-MATRICE, dacă referirea celui de-al doilea element al liniei 4 se realizează prin doi indecși, L și C, adică:

A (L, C)

valorile acestora trebuie să fie 48 și, respectiv, 4.

Trebuie precizat însă faptul că, în cadrul programelor, indecșilor li se atribuie valori (numai prin instrucțiunile SET, SEARCH și PERFORM VARYING) ca și cum ar reprezenta indici.

Există două modalități de referire a clementelor unui tablou utilizând indecși:

— *directă*, când se indică indexul (care reprezintă un identificator COBOL), ca de exemplu:

NOTA (I)

— *relativă*, când se indică indexul urmat de unul din operatorii aritmetici + sau - și de un literal (care reprezintă un număr întreg), ca de exemplu:

NOTA (J + 2)

NOTA (J - 1)

Utilizarea indecșilor este recomandată întrucât simplifică adresarea elementelor.

## 2. DESCRIEREA TABLOURILOR

Definirea unui tablou necesită precizarea următoarelor informații: categoria elementelor, dimensiunile acestuia, numărul de elemente ce corespund fiecărei dimensiuni și, eventual, cheile după care sînt ordonate elementele și indecșii asociați fiecărei dimensiuni. Categoria este precizată prin clauzele PICTURE și/sau USAGE, clauze care trebuie să figureze numai în rubricile de descriere a elementelor ce corespund ultimei dimensiuni. Celelalte informații sînt precizate prin intermediul clauzei OCCURS.

Format general:

OCCURS { *întreg-1* TIMES  
          { *întreg-1* TO *întreg-2* TIMES DEPENDING ON *nume-dată-1* }  
          { { ASCENDING } KEY IS *nume-dată-2* [, *nume-dată-3*] ... }  
          [INDEXED BY *index-1* [*index-2*] ...]



a. Opțiunea *întreg-1 TIMES* este utilizată pentru definirea tablourilor cu număr fix de elemente; acest număr este indicat prin *întreg-1*, care trebuie să fie un număr întreg pozitiv, mai mare decât zero. Clauza OCCURS cu opțiunea *întreg-1 TIMES* poate figura în maximum trei rubrici de descriere succesive (ale căror numere de nivel, care au valori în intervalul [2, 49], notate cu *n1*, *n2* și *n3*, satisfac relația:  $n1 < n2 < n3$ ), permițând definirea tablourilor cu maximum trei dimensiuni. Rezultă că o clauză OCCURS care figurează în rubrica de descriere a unei date, indică faptul că data respectivă reprezintă elementele ce corespund unei dimensiuni a tabloului. De exemplu, descrierea tablourilor cu număr fix de elemente din figura XVII.1 se poate face astfel:

01	TABLOU-NOTE.	
05	NOTA	PIC 99V99 OCCURS 8.
01	TABLOU-MATRICE.	
05	LINIE	OCCURS 4.
10	A	PIC S99V99 OCCURS 4.
01	TABLOU-PRODUCȚIE.	
05	SEMESTRU	OCCURS 2.
10	TRIMESTRU	OCCURS 2.
15	LUNA	OCCURS 3 PIC 9(5).

b. Opțiunea *întreg-2 TO întreg-3 TIMES* poate fi utilizată pentru definirea tablourilor cu număr variabil de elemente (aceste tablouri au o singură dimensiune); *întreg-2* (care poate avea și valoarea zero) indică numărul minim de elemente, *întreg-3*, numărul maxim de elemente, iar valoarea datei *nume-dată-1* (care nu trebuie să depășească valoarea lui *întreg-3*) indică numărul exact de elemente într-un moment al prelucrării. Evident, *întreg-2* și *întreg-3* trebuie să reprezinte numere întregi pozitive și să satisfacă relația:  $\text{întreg-2} < \text{întreg-3}$ ; de asemenea, data *nume-dată-1* trebuie să fie o dată numerică elementară, ale cărei valori reprezintă numere întregi pozitive.

Această opțiune a clauzei OCCURS permite descrierea articolelor de format variabil. De exemplu, articolul ART din figura XVII.1 poate fi descris astfel:

01	ART.	
05	NR-MATRICOL	PIC 9(5).
05	NUMAR-REPERE	PIC 9(2).
05	REPER	OCCURS 2 TO 10 DEPENDING ON NUMAR- REPERE.
10	NR-OPERATII	PIC 9(4).
10	COD-REPER	PIC 9(5).

În figura XVII.2 sînt prezentate trei articole de acest tip.

	REPER		REPER		REPER	
	NR-OPERATII	COD-REPER	NR-OPERATII	COD-REPER	NR-OPERATII	COD-REPER
Articol 1	00001	0010	0010	01000	0040	02000
Articol 2	00002	0020	0020	04000	0050	05000
Articol 3	00003	0070	0070	01000	09000	

Fig. XVII.2



c. Opțiunile **ASCENDING / DESCENDING** indică faptul că elementele tabloului sînt ordonate strict crescător / descrescător după valorile cheilor *nume-dată-2*, *nume-dată-3* ... Aceste chei pot fi descrise în rubrica în care figurează clauza **OCCURS** sau într-o rubrică subordonată acesteia. În cazul în care există mai multe chei, prioritatea lor este indicată în ordine descrescătoare (adică cheia *nume-dată-2* are prioritatea cea mai mare). Considerînd că elementele tabloului **TABLOU-REPERE** sînt ordonate crescător după valorile datei **COD-REPER**, descrierea acestuia se poate face astfel:

01 ART.

05 NR-MATRICOL	PIC 9(5).
05 NUMAR-REPERE	PIC 9(2).
05 REPER	OCCURS 2 TO 10 DEPENDING ON NUMAR-REPERE ASCENDING KEY IS COD-REPER.
10 NR-OPERATII	PIC 9(4).
10 COD-REPER	PIC 9(5).

d. Opțiunea **INDEXED BY** poate fi utilizată pentru a asocia dimensiunii unui tablou unul sau mai mulți indecși. Un *index reprezintă* o dată care nu trebuie descrisă printr-o rubrică aparte, compilatorul asociindu-i, implicit, o zonă de 4 locații. Asocierea de indecși dimensiunilor tablourilor, impune, în general, ca aceștia să fie utilizați pentru referirea elementelor ce corespund acestor dimensiuni.

De exemplu, dacă tabloul **TABLOU-MATRICE** este descris astfel:

01 TABLOU-MATRICE.

05 LINIE	OCCURS 4 INDEXED BY I.
10 A	OCCURS 4 PIC S99V99 INDEXED BY J.

înseamnă că dimensiunii ce corespunde datelor **LINIE** i-a fost asociat indexul **I**, iar celei ce corespunde datelor **A**, indexul **J**.

#### Observații

Clauza **OCCURS** nu poate figura într-o rubrică de descriere cu număr de nivel 01.

Un tablou cu număr variabil de elemente trebuie să fie descris ultimul în cadrul unui articol.

Privită în corelație cu clauzele **VALUE** și **REDEFINES**, utilizarea clauzei **OCCURS** are două restricții:

- elementelor unui tablou nu li se pot atribui valori prin clauza **VALUE**;
- un tablou cu număr variabil de elemente nu poate fi redefinit.

### 3. INSTRUCȚIUNI PENTRU PRELUCRAREA TABLOURILOR

Trei instrucțiuni pot fi utilizate pentru prelucrarea tablourilor: **PERFORM VARYING**, **SEARCH** și **SET**. Primele două descriu cicluri de prelucrare, iar ultima, operații de atribuire de valori indecșilor și datelor numerice.



a. Instrucțiunea **PERFORM VARYING** permite descrierea ciclurilor cu variabilă de control.

● În forma cea mai simplă, instrucțiunea are următorul format:

**PERFORM** *nume-prelucrare-1* [THRU *nume-prelucrare-2*]

**VARYING** *c* **FROM** *i* **BY** *p* **UNTIL** *Condiție*

unde:

— *nume-prelucrare-1* [THRU *nume-prelucrare-2*] precizează secvența care reprezintă corpul ciclului;

— *c* reprezintă variabila de control, care poate fi un index sau o dată numerică ale cărei valori trebuie să fie numere întregi pozitive;

— *i* reprezintă valoarea inițială a variabilei de control, care poate fi valoarea unui literal numeric întreg, a unui index sau a unei date numerice;

— *p* reprezintă valoarea (*pasul*) cu care este mărită, la fiecare execuție (*iterație*) a corpului ciclului, variabila de control, și poate fi valoarea unui literal numeric întreg sau a unei date numerice.

— *condiție* reprezintă o condiție simplă sau compusă, prin a cărei testare se stabilește continuarea execuției corpului ciclului sau ieșirea din ciclu.

Principiul de execuție al instrucțiunii **PERFORM VARYING** este prezentat în figura XVII.3. Se remarcă faptul că ieșirea din ciclu are loc în momentul în care valoarea logică a condiției testate este „adevărat”. În cazul utilizării instrucțiunii **PERFORM VARYING** pentru prelucrarea tablourilor, valorile curente ale variabilei de control pot fi utilizate pentru referirea (adresarea) elementelor acestora.

● Pentru descrierea a două și, respectiv, a trei cicluri cu variabilă de control, incluse unul în altul, pot fi utilizate următoarele formate ale instrucțiunii **PERFORM VARYING**:

**PERFORM** *nume-prelucrare-1* [THRU *nume-prelucrare-2*]  
**VARYING** *c*<sub>1</sub> **FROM** *i*<sub>1</sub> **BY** *p*<sub>1</sub> **UNTIL** *condiție-1*  
**AFTER** *c*<sub>2</sub> **FROM** *i*<sub>2</sub> **BY** *p*<sub>2</sub> **UNTIL** *condiție-2*

**PERFORM** *nume-prelucrare-1* [THRU *nume-prelucrare-2*]  
**VARYING** *c*<sub>1</sub> **FROM** *i*<sub>1</sub> **BY** *p*<sub>1</sub> **UNTIL** *condiție-1*  
**AFTER** *c*<sub>2</sub> **FROM** *i*<sub>2</sub> **BY** *p*<sub>2</sub> **UNTIL** *condiție-2*  
**AFTER** *c*<sub>3</sub> **FROM** *i*<sub>3</sub> **BY** *p*<sub>3</sub> **UNTIL** *condiție-3*

Semnificația variabilelor de control (*c*<sub>1</sub>, *c*<sub>2</sub> și *c*<sub>3</sub>), a valorilor inițiale (*i*<sub>1</sub>, *i*<sub>2</sub> și *i*<sub>3</sub>), a valorilor de incrementare a variabilelor de control (*p*<sub>1</sub>, *p*<sub>2</sub> și *p*<sub>3</sub>), precum și a condițiilor testate (*condiție-1*, *condiție-2* și *condiție-3*) este aceeași ca în cazul primului format al instrucțiunii. Principiul de execuție a instrucțiunilor **PERFORM VARYING**, ce corespund formatelor 2 și 3, este prezentat în figurile XVII.4 și XVII.5.

Observația care se poate face referitor la ultimele două formate este aceea că secvența *nume-prelucrare-1* [THRU *nume-prelucrare-2*] este comună ciclurilor descrise prin instrucțiune. De exemplu, secvența din figura XVII.6, care determină și tipărește suma elementelor liniilor matricei A (4,4) nu poate fi descrisă printr-o singură instrucțiune **PERFORM VARYING** deoarece

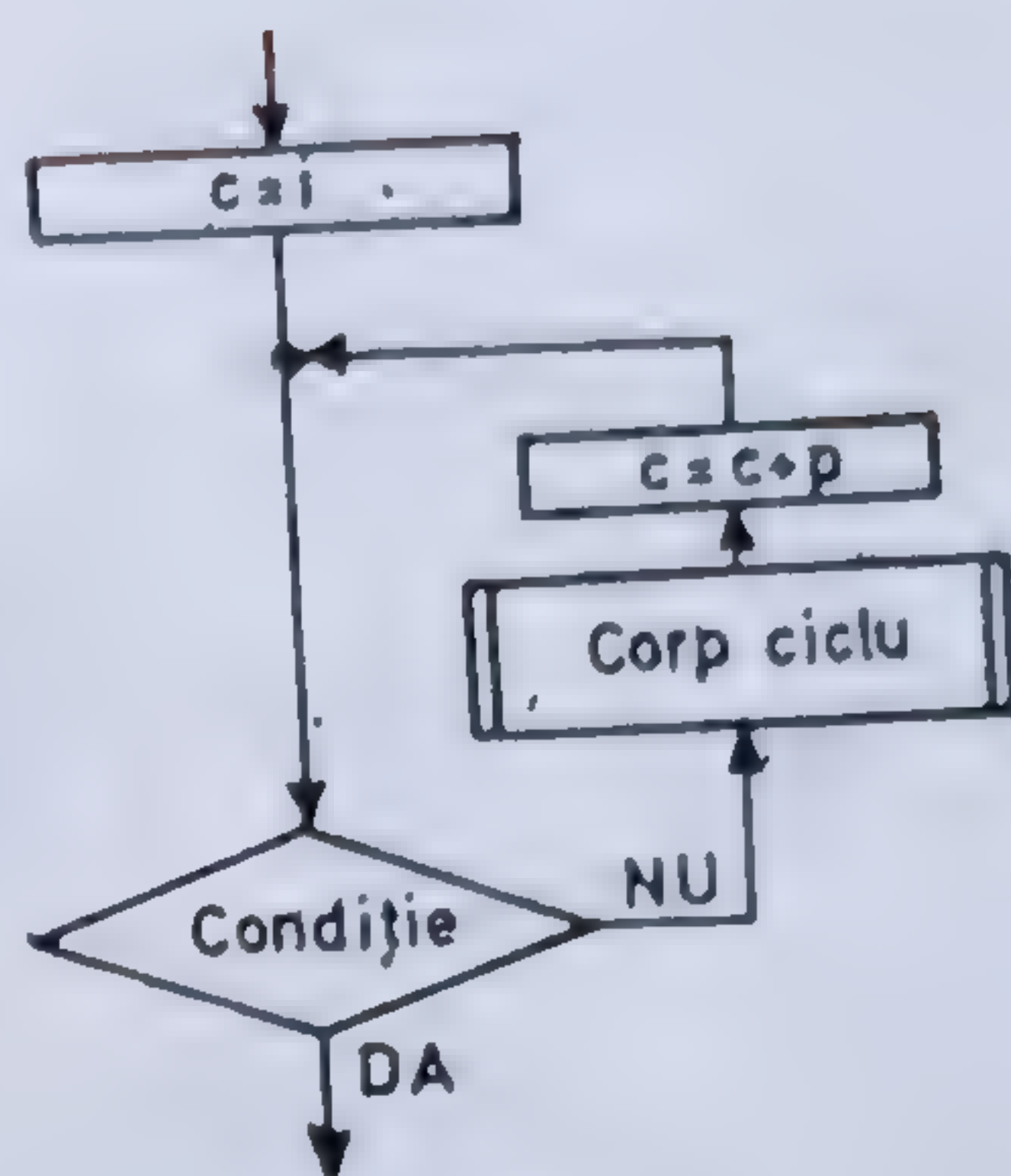


Fig. XVII.3



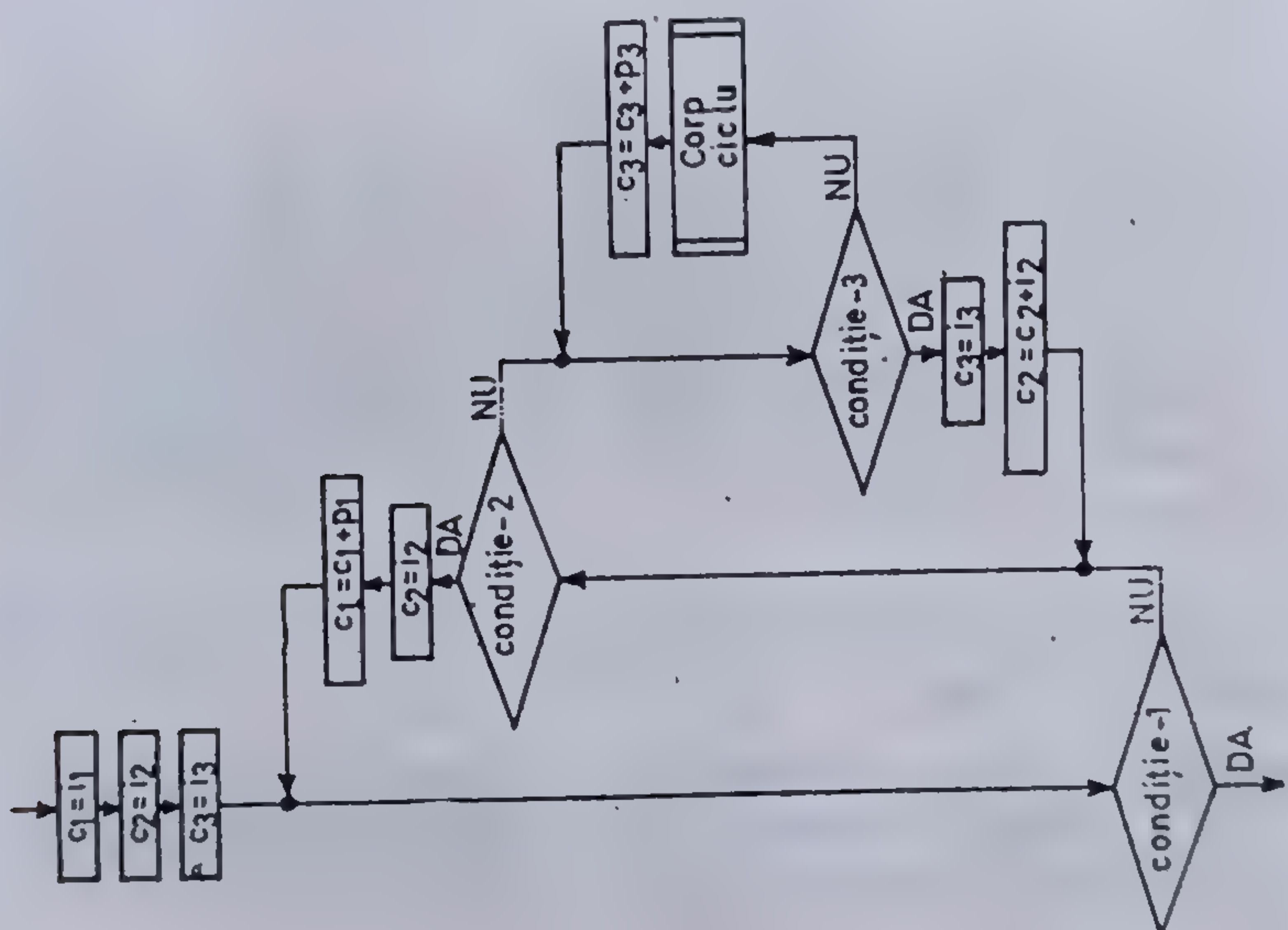


Fig. XVII.5

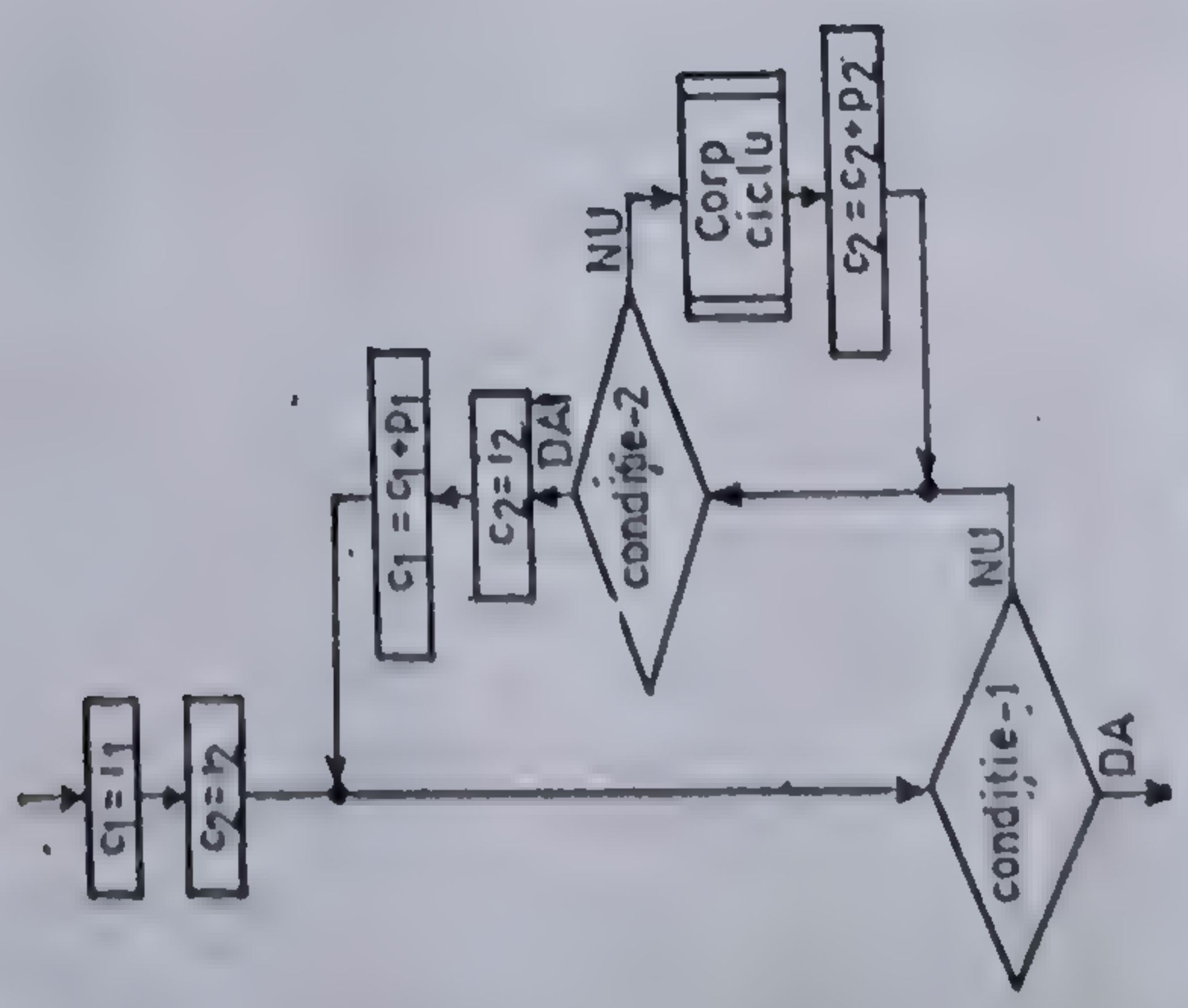


Fig. XVII.4

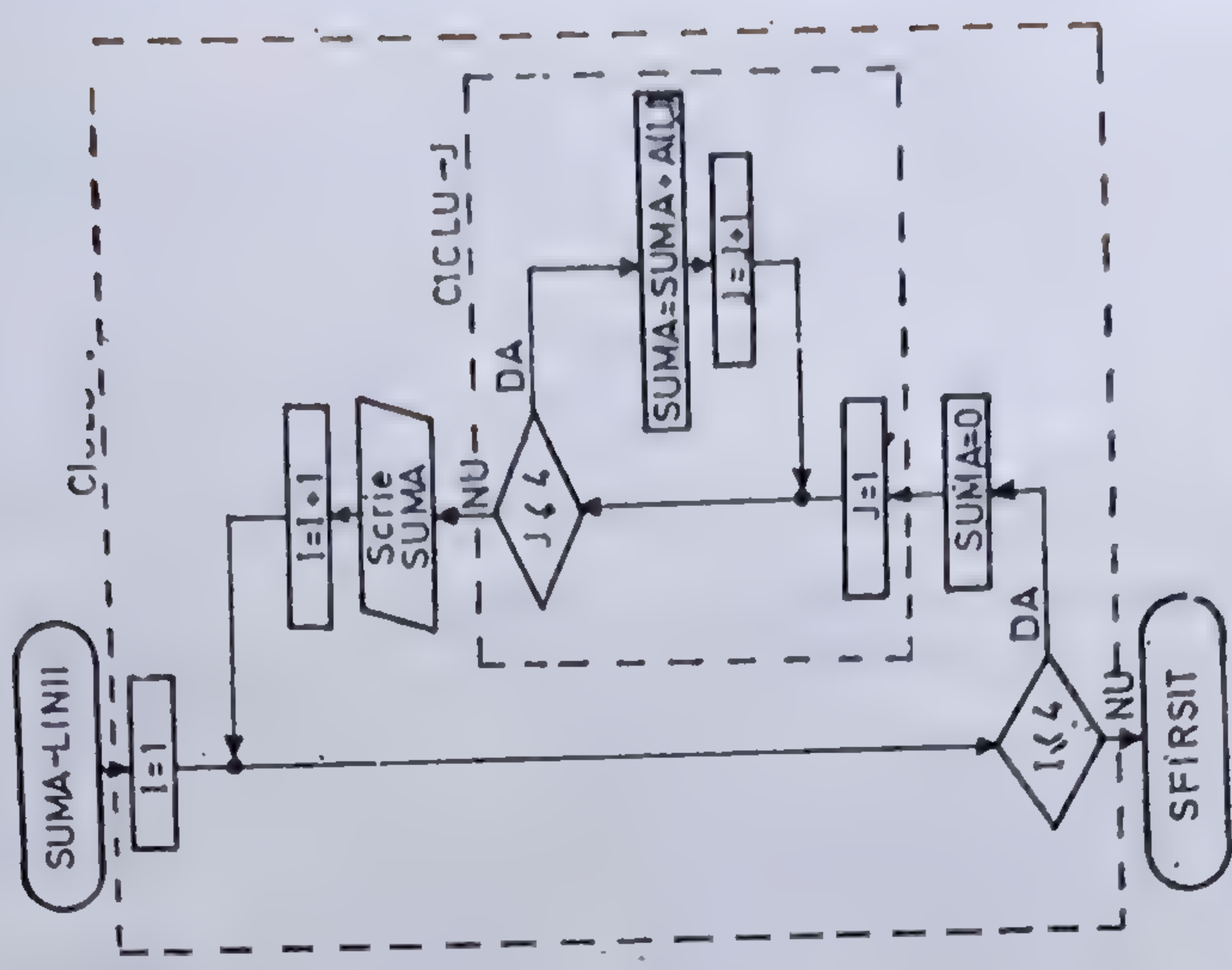


Fig. XVII.6



Între cele două cicluri controlate de variabilele I și J se interpun operațiile de inițializare și tipărire a valorilor datei SUMA. Această secvență poate fi descrisă astfel:

## SUMA-LINII.

```
PERFORM CICLU-I VARYING I FROM 1 BY 1 UNTIL I > 4
CICLU-I.
```

MOVE ZERO TO SUMA

PERFORM CICLU-J VARYING J FROM 1 BY 1 UNTIL J &gt; 4

DISPLAY 'SUMA ELEMENTELOR LINIEI 'I' ESTE 'SUMA.

## CICLU-1.

ADD A (I J) TO SUMA.

*Exemplu.* Fie un fișier pe cartele, ale cărui articole conțin date referitoare la producțiile lunare realizate în cadrul întreprinderilor unei centrale industriale: cod întreprindere (3 caractere numerice), denumire întreprindere (15 caractere alfanumerice), producția realizată în cele 12 luni ale anului (5 caractere numerice).

Se cere ca, prin consultarea acestui fișier, să se editeze la imprimantă un raport care să conțină, pentru fiecare întreprindere, un rînd care să indice codul întreprinderii și producțiile trimestriale. Macheta raportului este prezentată în figura XVII.7.

COD				TRIMESTRUL 1				TRIMESTRUL 2				TRIMESTRUL 3				TRIMESTRUL 4			
X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

**Fig. XVII.7**

Schema logică este prezentată în figura XVII.8, iar rezolvarea în programul XVII.1.

b. Instrucțiunea **SET** este utilizată pentru a atribui valori indecșilor și are două formate:

### Format general 1:

$$\text{SET } \left\{ \begin{matrix} \text{index-1} \\ \text{nume-dată-1} \end{matrix} \right\} \left[ , \left\{ \begin{matrix} \text{index-2} \\ \text{nume-dată-2} \end{matrix} \right\} \right] \dots \text{TO } \left\{ \begin{matrix} \text{index-3} \\ \text{nume-dată-3} \\ \text{literal-1} \end{matrix} \right\}$$

### Format general 2:

$$\text{SET } \textit{index-1} [, \textit{index-2}] \dots \left\{ \begin{array}{l} \text{UP BY} \\ \text{DOWN BY} \end{array} \right\} \left\{ \begin{array}{l} \textit{numb-datd-1} \\ \textit{literal-1} \end{array} \right\}$$

Principiul de execuție este, în general, următorul:

— valoarea indexului *index-3*, a datei *nume-dată-3* sau a literalului *literal-1* se atribuie indexului *index-1* sau datei *nume-dată-1*, indexului *index-2* sau datei *nume-dată-2* ș.a.m.d. — în cazul formatului 1;

— valoarea indexului *index-1*, a indexului *index-2* ș.a.m.d. este mărită (UP BY) sau micșorată (DOWN BY) cu valoarea datei *nume-dată-1* sau a literalului *literal-1* — în cazul formatului 2.

Indiferent de formatul utilizat, datele *nume-dată-1*, *nume-dată-3* trebuie să fie date elementare numerice, iar valorile lor, cât și ale literalului *literal-1*,



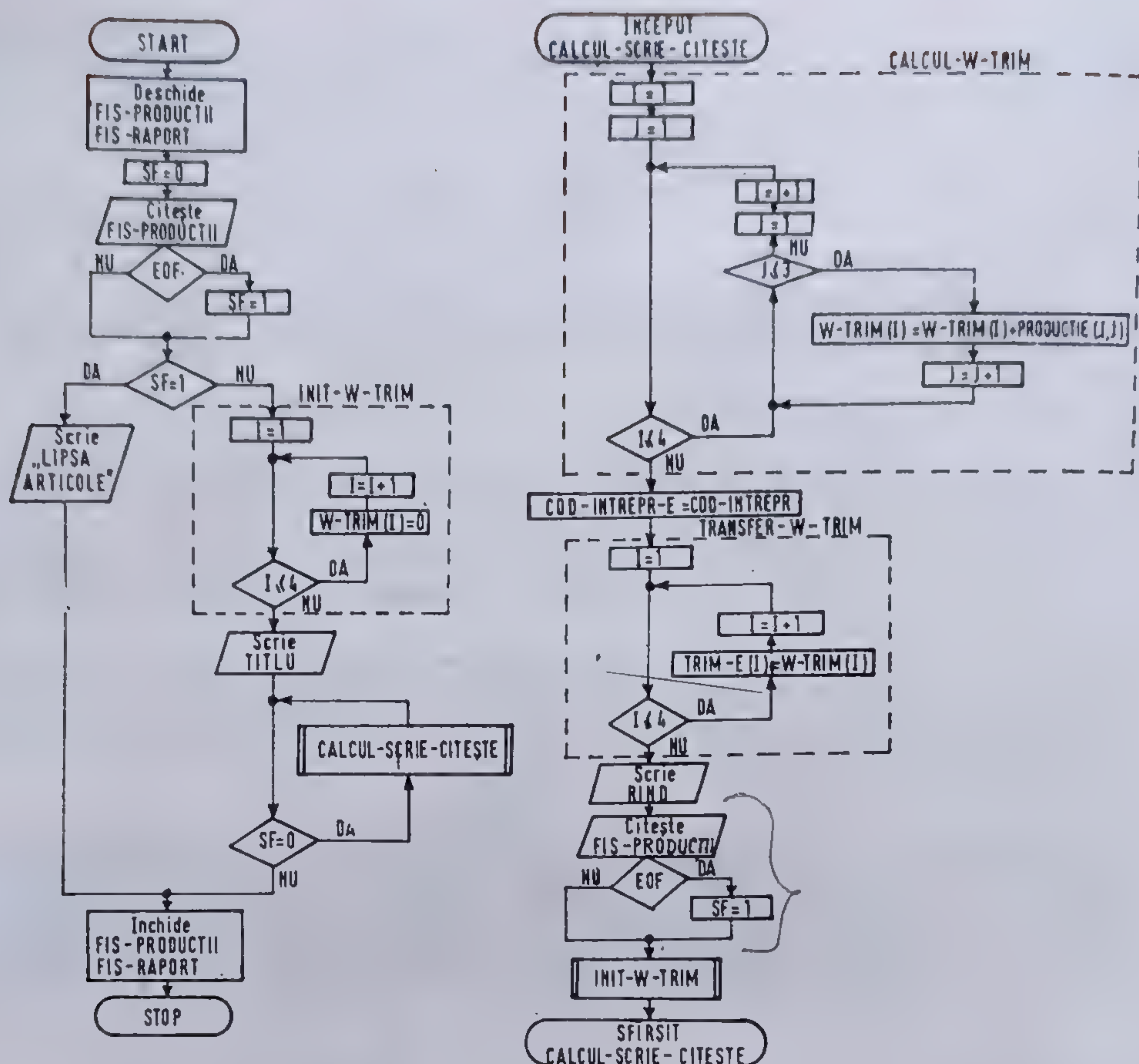


Fig. XVII.8

trebuie să fie numere întregi pozitive. În cazul formatului 1 nu pot fi descrise operații de atribuire de forma:

SET *nume-dată-1* [*nume-dată-2*] ... TO  $\left\{ \begin{array}{l} \textit{nume-dată-3} \\ \textit{literal-1} \end{array} \right\}$

întrucât ele pot fi descrise prin instrucțiunea MOVE. Rezultă că, într-o operație de atribuire descrisă prin instrucțiunea SET, trebuie să fie implicat cel puțin un index.

c. Instrucțiunea **SEARCH** descrie cicluri cu variabilă de control pentru căutarea, în cadrul tablourilor cu o dimensiune, a unui element care satisface o anumită condiție. Căutarea poate fi *secvențială* sau *binară* (dichotomică).

● În cazul căutării *secvențiale*, tabloul este explorat începând cu un element specificat; căutarea ia sfârșit la întâlnirea elementului care satisface condiția sau la terminarea parcurgerii tabloului.

Formatul instrucțiunii este următorul:

**SEARCH** *nume-dată-1* [VARYING *index*]  
 [;AT END *instrucțiune-1* ...]  
 ;WHEN *condiție-1* *instrucțiune-2* ...  
 [;WHEN *condiție-2* *instrucțiune-3* ...].



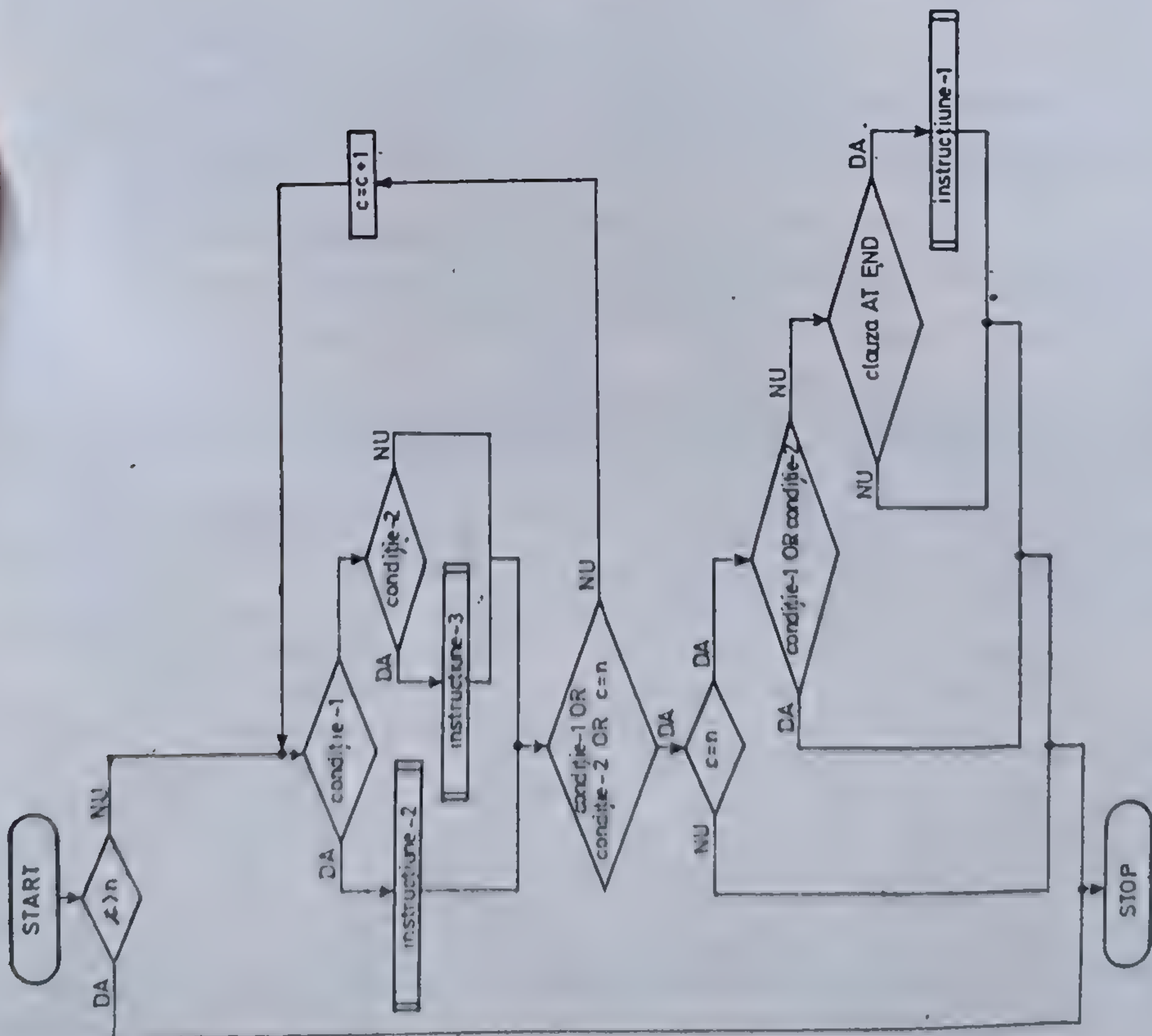


Fig. XVII.9

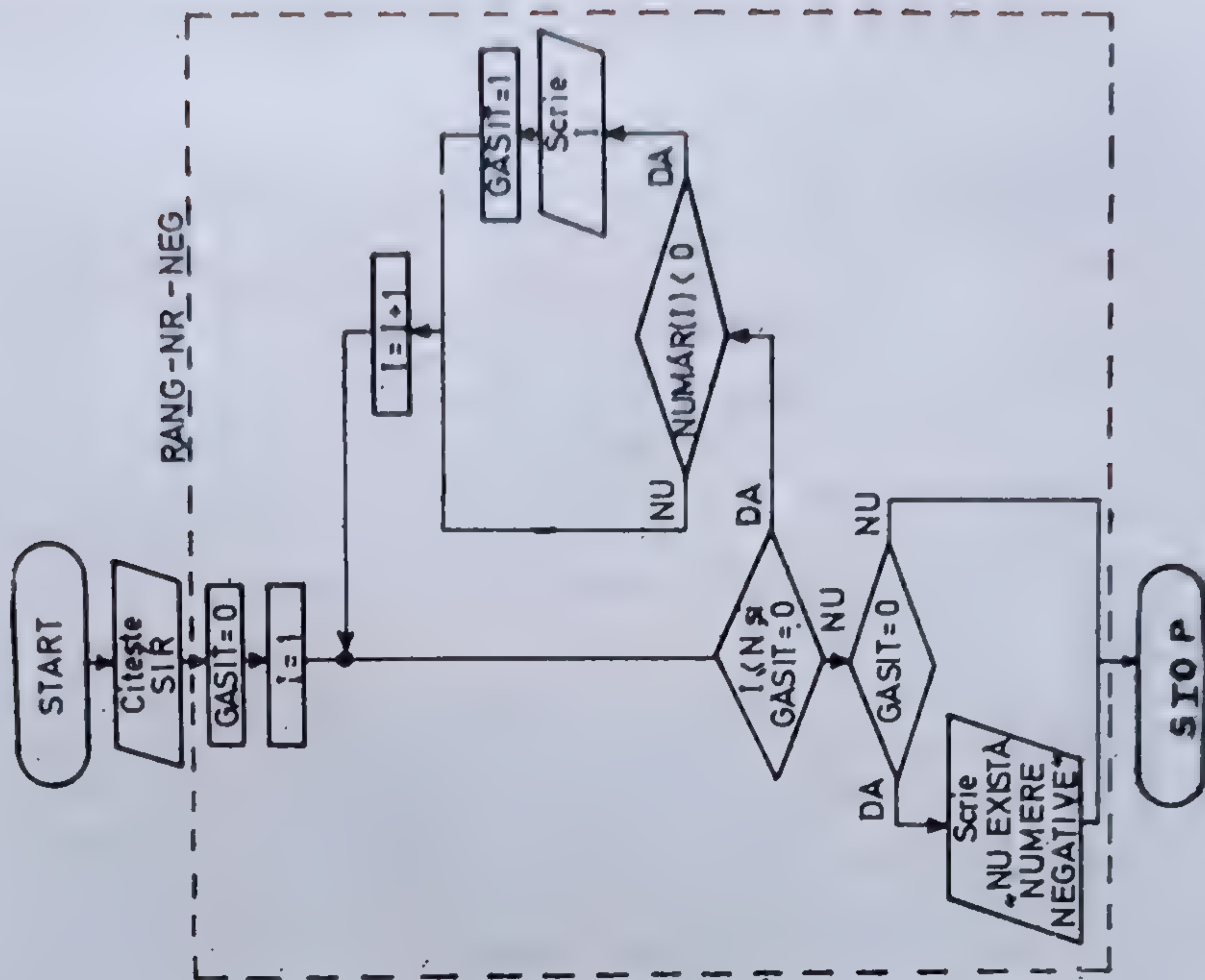


Fig. XVII.10



```

ID DIVISION.
PROGRAM-ID. PVII2.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 N PIC 99.
01 SIR.
   05 NUMAR PIC 99(5) OCCURS 3 TO 15
      DEPENDING ON N INDEXED BY I.
01 INDICI.
   05 GASIT PIC 9.
   05 INDICE PIC 99.
PROCEDURE DIVISION.
P-PRINCIPAL.
   ACCEPT N
   ACCEPT SIR
   PERFORM RANG-NR-NEG
   STOP RUN.
RANG-NR-NEG.
   MOVE ZERO TO GASIT
   SET I TO 1
   SEARCH NUMAR WHEN NUMAR (I) < ZERO
      SET INDICE TO 1
      MOVE 1 TO GASIT.
   IF GASIT = 0
      THEN
         DISPLAY 'STIPLU NU CONTINE NUMERE NEGATIVE'
      ELSE
         DISPLAY 'PRIMUL NUMAR NEGATIV DEDUPA PULITIA A
            INDICE A A'.

```

#### PROGRAMUL XVII.2

unde:

— *nume-dată-1* reprezintă numele tabloului; în rubrica de descriere în care apare identificatorul *nume-dată-1* trebuie să figureze clauzele OCCURS și INDEXED BY;

— *index* reprezintă unul dintre indecșii asociați tabloului;

— *instrucțiune-1* indică secvența de instrucțiuni care va fi executată dacă tabloul nu conține elementul căutat;

— *condiție-1, condiție-2 ...* indică testele ce vor fi efectuate asupra elementelor tabloului; aceste condiții pot fi simple sau compuse;

— *instrucțiune-2, instrucțiune-3 ...* reprezintă secvențele de instrucțiuni care vor fi executate dacă în tablou este găsit elementul pentru care valoarea logică a condiției *condiție-1, condiție-2 ...* este „adevărat” (condițiile sînt evaluate în ordinea în care apar în instrucțiune; de exemplu, dacă în instrucțiunea SEARCH sînt citate trei condiții, iar pentru un anumit element al tabloului valoarea logică a condiției *condiție-1* este „fals”, iar cea a condiției *condiție-2* este „adevărat”, va fi executată secvența indicată prin *instrucțiune-3*, condiția *condiție-3* nemaifiind evaluată).

Variabila de control a ciclului este primul index citat în rubrica de descriere a tabloului, în cazul în care clauza VARYING lipsește, sau indexul citat în



clauza VARYING; valoarea inițială a variabilei de control trebuie stabilită prin instrucțiunea SET.

În figura XVII.9 este prezentat principiul de execuție a instrucțiunii SEARCH, în cazul căutării secvențiale, considerînd situația în care sînt testate două condiții. Notățiile utilizate au următoarea semnificație:

$c$  — variabila de control al ciclului;

$n$  — numărul maxim de elemente ale tabloului explorat.

*Exemplu.* Se consideră un șir de  $n$  numere, citite de pe două cartele prin instrucțiunea ACCEPT;  $n$  poate avea o valoare cuprinsă între 3 și 15, iar fiecare număr este format din maximum 5 cifre.

Să se scrie un program care să determine și să tipărească rangul (numărul de ordine) primului număr negativ.

Rezolvarea este prezentată în programul XVII.2, conform schemei din figura XVII.10.

● În cazul căutării binare, elementele tabloului trebuie să fie ordonate (strict crescător sau descrescător) după valorile uneia sau mai multor chei, care trebuie indicate în rubrica de descriere a tabloului.

Pentru prezentarea metodei de căutare binară se consideră un tablou  $T$  ale cărui elemente  $e$ , în număr de  $n$ , sînt ordonate crescător după valorile unei chei  $c$ . Găsirea în acest tablou a elementului pentru care valoarea cheii  $c$  este egală cu  $v$  reprezintă un proces iterativ. Prima iterație constă în stabilirea unui indice  $i$ , după relația:  $i = \left\lfloor \frac{1+n}{2} \right\rfloor$ , și compararea valorii cheii elementului  $e(i)$  cu  $v$ . Rezultatul comparării poate fi:

—  $c(i) = v$  — elementul căutat a fost găsit;

—  $c(i) < v$  — elementul căutat se află în prima parte a tabloului, notată cu  $T_1$ , și care conține primele  $\left\lfloor \frac{n}{2} \right\rfloor$  sau  $\left\lfloor \frac{n}{2} \right\rfloor + 1$  elemente ale tabloului  $T$ , după cum  $n$  este impar sau par;

—  $c(i) > v$  — elementul căutat se află în a doua parte a tabloului, notată cu  $T_2$ , și care conține ultimele  $\left\lfloor \frac{n}{2} \right\rfloor + 1$  sau  $\left\lfloor \frac{n}{2} \right\rfloor$  elemente ale tabloului.

În cazul în care elementul căutat se află în subtabloul  $T_1$ , valoarea indicelui primului element al acestuia rămîne 1, iar cea a ultimului element devine  $i$ ; în cazul în care elementul căutat se află în subtabloul  $T_2$ , valoarea indicelui primului element al acestuia devine  $i$ , iar cea a ultimului element, rămîne  $n$ .

În funcție de subtabloul în care se află elementul căutat, a doua iterație — care se efectuează numai dacă  $c(i) < v$  sau  $c(i) > v$  — constă în stabilirea unui nou indice  $i'$ , după relația:

$$i' = \left\lfloor \frac{1+i}{2} \right\rfloor$$

sau, respectiv,

$$i' = \left\lfloor \frac{i+n}{2} \right\rfloor$$

compararea cheii elementului  $e(i)$  cu  $v$  ș.a.m.d.

Pentru a ilustra concret această metodă se consideră exemplul prezentat în continuare.



Fie un tablou ale cărui elemente conțin numele și mediile generale ale elevilor unei clase (fig. XVII.11); elementele sînt ordonate crescător după valorile datei NUME. Se cere să se găsească în acest tablou elementul ce corespunde elevului POPESCU și să se tipărească media generală. O modalitate de rezolvare este prezentată în figura XVII.12.

TABLLOU

ELEV		ELEV	
NUME	MEDIA	NUME	MEDIA
X(25)	99V99	X(25)	99V99

ELEV	
NUME	MEDIA
X(25)	99V99

Fig. XVII.11

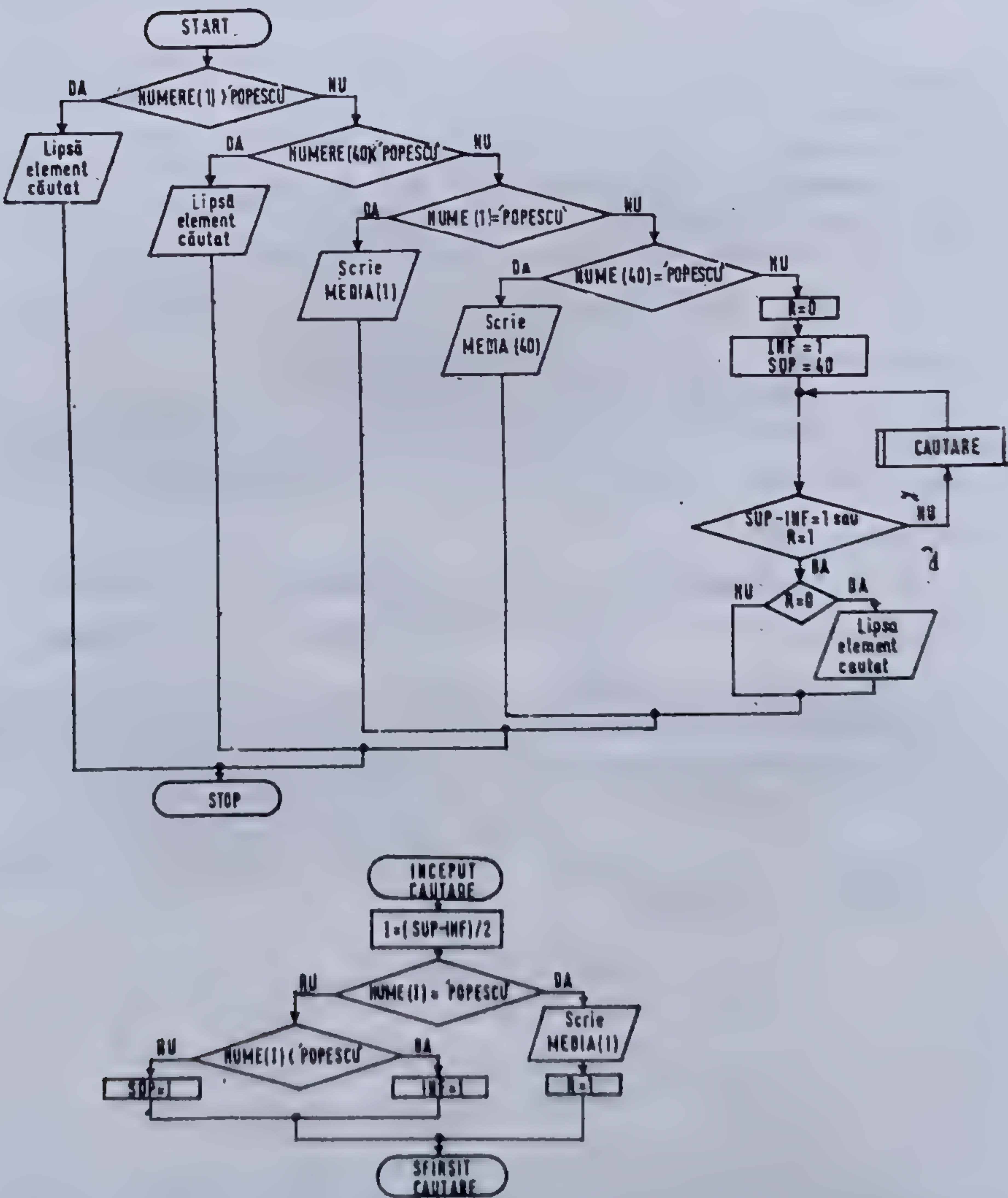


Fig. XVII.12



Formatul general al instrucțiunii SEARCH, în cazul căutării binare, este:

```
SEARCH ALL nume dată  
      [;AT END instrucțiune-1 ...]  
      ;WHEN condiție instrucțiune-2 ...
```

unde:

— *nume-dată* reprezintă numele tabloului prelucrat; în rubrica de descriere în care figurează identificatorul *nume-dată* trebuie să figureze clauzele OCCURS, INDEXED BY și KEY;

— *instrucțiune-1* indică secvența de instrucțiuni care va fi executată dacă în tablou nu este găsit elementul căutat;

— *condiție* poate reprezenta: o condiție simplă de tip test de relație — relația testată fiind „egal” — sau de tip test de nume de condiție; o condiție compusă formată din condiții simple de tip test de relație și test de nume de condiție, unite prin operatorul logic AND (pot fi testate însă numai valorile datelor citate în clauza KEY);

— *instrucțiune-2* indică secvența de instrucțiuni ce va fi executată în cazul în care în tablou este găsit un element pentru care valoarea logică a condiției *condiție* este „adevărat”.

În cazul exemplului prezentat anterior, dacă în secțiunea WORKING sînt descrise următoarele articole:

01 TABLOU.	
05 ELEV	OCCURS 40 INDEXED BY I ASCENDING KEY NUME.
10 NUME	PIC X(25).
10 MEDIA	PIC 99V99.
01 ARTICOL.	
05 MEDIA-E	PIC 99.99.
05 R	PIC 9 VALUE ZERO.

instrucțiunea SEARCH poate fi scrisă astfel:

```
SEARCH ALL ELEV  
      AT END MOVE 1 TO R  
      WHEN NUME = 'POPESCU'  
          MOVE MEDIA (I) TO MEDIA-E  
          DISPLAY MEDIA-E.  
      IF R = ZERO  
      THEN  
          DISPLAY 'LIPSĂ POPESCU'.
```

## PROBLEME

1. Există diferență între rezultatele obținute prin execuția secvențelor următoare:

a)        MOVE 0 TO SUMA  
          MOVE 0 TO I  
          PERFORM ADUNA  
                  UNTIL I > 10



## ADUNA

ADD 1 TO I  
ADD X (I) TO SUMA.

b) MOVE 0 TO SUMA  
PERFORM ADUNA  
VARYING I FROM 1 BY 1  
UNTIL I = 10

·  
·  
·

## ADUNA.

ADD X (I) TO SUMA.

c) MOVE 10 TO I  
PERFORM ADUNA I TIMES.

·  
·  
·

## ADUNA.

ADD X (I) TO SUMA  
SUBTRACT 1 FROM I.

2. Să se scrie un program care să realizeze conversia unui număr (format din maximum 5 cifre) din baza 10 în baza 2.
3. Să se scrie un program care să realizeze înmulțirea a două matrici A(10,15) și B(15,7) și tipărirea matricei rezultat. Elementele celor două matrici sînt citite de pe cartele, fiecare cartelă conținînd elementele unei linii.
4. Se consideră un fișier pe cartele ale cărui articole conțin informații referitoare la producțiile lunare realizate de întreprinderile unei centrale industriale (vezi structura articolului fișierului FIS-PRODUCȚII din programul XVII.1).  
Să se scrie un program care să tipărească la imprimantă, pentru fiecare întreprindere, luna (în clar) în care s-a realizat o producție mai mare decît 10000. (Pentru căutarea în tablou a valorii respective se va utiliza instrucțiunea SEARCH).



## CAPITOLUL XVIII

### PRELUCRAREA FIȘIERELOR

#### 1. NOȚIUNI DESPRE FIȘIERE

a. **Generalități.** Caracteristica de bază a limbajului COBOL este aceea că permite descrierea prelucrărilor în care volumul de date este mare. Pentru realizarea acestor prelucrări, datele sînt grupate în fișiere.

Unitatea elementară de organizare și prelucrare a unui fișier se numește *înregistrare*. Deoarece organizarea unui fișier poate fi considerată la nivel logic sau la nivel fizic, înregistrările ce corespund acestor nivele se numesc *înregistrări logice* (sau *articole*) respectiv, *înregistrări fizice*.

O înregistrare fizică poate fi formată din mai multe înregistrări logice sau poate conține doar o parte a unei înregistrări logice. În general, în cazul suporturilor nereutilizabile înregistrarea logică se identifică cu înregistrarea fizică, iar în cazul suporturilor reutilizabile o înregistrare fizică conține mai multe înregistrări logice.

Pentru fișierele înregistrate pe suporturi reutilizabile apare necesitatea identificării și delimitării lor, întrucît pe o unitate de suport (rolă de bandă, pachet de discuri) pot fi înregistrate mai multe fișiere. În acest scop, acestor fișiere li se asociază informații de control, numite *etichete*.

Etichetele pot avea conținut și structură standard, caz în care sînt create prin proceduri ale sistemului de operare, sau conținut și structură oarecare, caz în care sînt create prin proceduri stabilite de utilizator. Etichetele din prima categorie se numesc *etichete standard*, iar cele din categoria a doua, *etichete utilizator*.

Limbajul COBOL dispune de clauze și instrucțiuni care permit descrierea atît a caracteristicilor fișierelor cît și a operațiilor la care acestea pot fi supuse; aceste operații sînt realizate efectiv prin intermediul sistemului de operare și anume de componenta acestuia — Sistemul de Gestiune a Fișierelor (SGF).

b. **Sistemul de gestiune a fișierelor.** Principalele funcții realizate de SGF se referă la: afectarea de suport fișierelor; blocarea și deblocarea articolelor; gestiunea zonelor-tampon.

● Prin **afectare**, fișierelor li se atribuie cantitatea de suport necesară înregistrării articolelor care le compun. În cazul suporturilor nereutilizabile afectarea este definitivă, fiind realizată de utilizator. În cazul suporturilor reutilizabile, afectarea poate fi definitivă sau temporară, fiind realizată de SGF și monitor.

● **Formatul articolelor** unui fișier poate fi:  
— *fix*, cînd toate articolele au aceeași lungime;  
— *variabil*, cînd lungimea diferă de la un articol la altul; în cazul suporturilor reutilizabile lungimea este asociată articolelor, în momentul înregistrării lor pe suport;

— *nedefinit*, cînd lungimea diferă de la un articol la altul, însă nu este asociată articolelor pe suport.

În cazul fișierelor pe suporturi reutilizabile cu articole de format fix sau variabil, înregistrările fizice pot conține mai multe articole. Operația



de constituire a înregistrărilor fizice se numește **blocare** (grupare) și are loc când fișierul este creat (scris); la consultarea (citirea) fișierului are loc operația inversă — **deblocarea** — care constă în separarea articolelor ce compun înregistrările fizice, în vederea prelucrării.

● Realizarea operațiilor de intrare/ieșire (citire/scriere) implică existența unei zone de memorie, numită **zonă-tampon**, în care sînt memorate înregistrările fizice ale fișierului, înainte de scrierea lor pe suport sau, după citirea lor de pe suport. Lungimea acestei zone este funcție de lungimea înregistrărilor fizice. Pentru realizarea unei simultaneități între prelucrarea datelor ce compun articolele înregistrărilor fizice ale unui fișier și realizarea operațiilor de intrare/ieșire, acestuia îi pot fi afectate mai multe zone-tampon.

Articolele unui fișier pot fi prelucrate direct în zona-tampon, sau într-o zonă distinctă, numită **zonă-articol**, rezervată de compilator implicit. Lungimea zonei-articol este funcție de lungimea articolelor fișierului. În primul caz se spune că modul de acces la articole este *MOVE*, iar în al doilea caz, *LOCATE*. Modul de acces *LOCATE* este utilizat, în general, când articolele fișierului sînt transferate, fără a suferi modificări, într-un alt fișier. Modul de acces *MOVE*, deși implică rezervarea unei zone de memorie suplimentară (zona-articol), este frecvent utilizat deoarece zona-articol conține numai datele proprii ale articolelor nu și cele ce le sînt asociate de SGF (de exemplu, lungimea în cazul articolelor de format variabil) în vederea memorării și înregistrării pe suport, simplificîndu-se în acest fel modul de adresare a datelor ce compun articolele.

c. **Moduri de acces la înregistrările fișierelor.** Modul de acces la articolele unui fișier reprezintă modalitatea prin care acestea sînt scrise/citite pe/de pe suportul afectat fișierului. Există două moduri de acces:

— *secvențial*, când un articol, al cărui număr de ordine în cadrul fișierului este  $n$ , este scris/citit numai după scrierea/citirea celor  $n - 1$  articole care îl preced pe suport;

— *direct*, când fiecare articol al fișierului poate fi scris/citit în/din cadrul unei anumite unități adresabile a suportului.

d. **Moduri de organizare a fișierelor.** Modul de organizare a unui fișier reprezintă metoda de dispunere (înregistrare) a articolelor acestuia pe suport, astfel încît să poată fi regăsite. Trei moduri de organizare a fișierelor pot fi definite în COBOL: *secvențială*, *secvențială-indexată* și *selectivă* (directă).

În cazul organizării *secvențiale*, articolele sînt înregistrate pe suport unele după altele, iar singurul mod de acces permis este cel secvențial.

În cazul organizării *secvențiale-indexate*, articolele, care sînt identificabile în mod unic după valorile unei chei, sînt înregistrate pe suport secvențial, în ordinea crescătoare a valorilor cheii și sînt repertoriate (indexate) într-o tabelă, numită *tabelă de indecși*. Regăsirea unui articol se poate realiza atît în acces secvențial cît și în acces direct; accesul secvențial presupune citirea tuturor articolelor care preced articolul respectiv în fișier, iar accesul direct necesită furnizarea valorii cheii articolului căutat și se realizează prin consultarea tabelii de indecși care, pe baza acestei valori, va indica adresa unității logice de suport în care se află articolul.

În cazul organizării *selective*, un articol al fișierului, care este identificabil în mod unic după valoarea unei chei, este înregistrat pe suport într-o zonă alocată unei clase (unui grup) de articole. Adresa acestei zone este stabilită printr-un algoritm (calcul) aplicat, de regulă, valorii cheii de identificare a articolelor. Regăsirea acestui articol în fișier se realizează indicînd valoarea



cheii de identificare și clasa din care face parte. Atât înregistrarea cât și regăsirea articolelor se realizează în acces direct.

e. **Tipuri de prelucrări ale fișierelor.** În COBOL asupra unui fișier pot fi efectuate trei tipuri de prelucrări: creare (scriere), consultare (citire) și actualizare (citire/scriere). *Crearea* constă în înregistrarea articolelor pe suportul afectat fișierului. *Consultarea* constă în transferul articolelor, de pe suportul pe care sînt înregistrate, în memorie, în vederea prelucrării lor. Prin actualizare, fișierului îi sînt *adăugate* noi articole, unele articole sînt *șterse*, iar altele sînt *modificate*; efectuarea uneia dintre aceste trei operații — adăugare, ștergere, modificare — presupune, în cazul fișierelor de organizare secvențială, citirea fișierului.

## 2. DESCRIEREA FIȘIERELOR ÎN LIMBAJUL COBOL

Precizarea caracteristicilor unui fișier (tipul unității periferice care îi este afectată, modul de organizare, modul de acces, formatul și structura articolelor, caracteristicile datelor ce compun articolele) se realizează prin intermediul rubricilor de descriere din secțiunile INPUT-OUTPUT și FILE.

a. **Paragraful FILE-CONTROL.** În acest paragraf fiecărui fișier îi este asociată o rubrică SELECT care precizează, prin intermediul unei clauze de descriere, caracteristicile generale ale fișierului. Formatul general al rubricii SELECT este următorul:

**SELECT** *nume-fișier*

**ASSIGN TO**  $\left\{ \begin{array}{l} \text{index} \\ \text{SYSIN} \\ \text{SYSOUT} \\ \text{SYSPUNCH} \end{array} \right\}$

$\left[ \text{,ORGANIZATION MODE IS} \left\{ \begin{array}{l} \text{SEQUENTIAL} \\ \text{INDEXED} \\ \text{DIRECT} \end{array} \right\} \right]$

$\left[ \text{,ACCESS MODE IS} \left\{ \begin{array}{l} \text{SEQUENTIAL} \\ \text{RANDOM} \end{array} \right\} \right]$

$\left[ \text{,RESERVE} \left\{ \begin{array}{l} \text{intreg} \\ \text{NO} \end{array} \right\} \text{ALTERNATE} \left\{ \begin{array}{l} \text{AREA} \\ \text{AREAS} \end{array} \right\} \right]$

Singurele clauze obligatorii sînt SELECT și ASSIGN; celelalte sînt opționale și pot apărea în orice ordine în cadrul rubricii SELECT.

● **Clauza SELECT** precizează numele care îi este asociat fișierului; prin identificatorul *nume-fișier* fișierul poate fi referit în cadrul programului.

● **Clauza ASSIGN** precizează identificatorul de exploatare și tipul unității periferice afectată fișierului. În cazul în care fișierului îi este afectată o unitate standard a sistemului, identificatorul de exploatare și tipul unității periferice sînt indicate prin unul dintre numele simbolice SYSIN, SYSOUT sau SYSPUNCH.



În cazul în care fișierului nu îi este afectată o unitate standard a sistemului, identificatorul de exploatare este indicat printr-o literă, urmată de două caractere care specifică tipul unității periferice:

- CR — cititor de cartele;
- PR — imprimantă;
- CP — perforator de cartele;
- MT — unitate de bandă magnetică;
- MD — unitate de discuri DIMAS (MD 25);
- AD — unitate de discuri DIMAS (MD 50);
- BD — unitate de discuri DIMAS (MD 100);
- CD — unitate de discuri DIMAS (MD 200);
- DD — unitate de discuri DIMAS (MD 400);
- DK — unitate de discuri de tip oarecare.

● *Clauza ORGANIZATION* indică, prin opțiunile SEQUENTIAL (opțiune implicită), INDEXED și DIRECT, modul de organizare a fișierului: organizare secvențială, organizare secvențială-indexată și, respectiv, organizare selectivă.

● *Clauza ACCESS* indică, prin opțiunile SEQUENTIAL (opțiune implicită) și RANDOM, modul de acces la articolele fișierului: acces secvențial și, respectiv, acces direct.

● *Clauza RESERVE* permite utilizatorului să indice suplimentarea sau micșorarea numărului de zone-tampon rezervate implicit de compilator unui fișier. În absența acestei clauze, compilatorul rezervă două zone-tampon fișierelor secvențiale și secvențiale-indexate și o zonă-tampon fișierelor selective. În cazul în care este utilizată opțiunea *întreg*, numărul de zone-tampon este suplimentat cu valoarea lui *întreg*, dacă aceasta este cuprinsă între 1 și 12, sau limitat la 1, dacă aceasta este zero. În cazul în care este utilizată opțiunea NO compilatorul rezervă fișierului o singură zonă-tampon.

b. **Paragraful I-O CONTROL.** În acest paragraf sînt precizate tehnicile speciale utilizate în prelucrarea fișierelor.

Format general:

#### **I-O-CONTROL.**

[**APPLY LOCATE MODE ON** *nume-fișier-1* [*,nume-fișier-2*] ...]

[**;****SAME** [**RECORD**] **AREA FOR** *nume-fișier-3* [*,nume-fișier-4*] ...]

[**;****MULTIPLE FILE CONTAINS** *nume-fișier-5* [*,nume-fișier-6*] ...].

Toate clauzele acestui paragraf sînt opționale; ele pot apărea în orice ordine în cadrul paragrafului.

● *Clauza APPLY* precizează fișierele pentru care modul de acces la articole este LOCATE. Acestor fișiere compilatorul nu le rezervă zone-articol, prelucrarea articolelor realizîndu-se direct în zonele-tampon. Clauza nu poate figura în descrierea fișierelor selective și a fișierelor secvențiale-indexate care sînt create sau actualizate, precum și a fișierelor cărora le sînt afectate unitățile standard ale sistemului. Prezența acestei clauze este însă obligatorie în descrierea fișierelor cu articole de format nedefinit.



### Observație

Prelucrarea în mod LOCATE impune anumite restricții în ceea ce privește cadrul datelor în cadrul articolelor. Astfel, dacă articolele fișierelor prelucrate în mod LOCATE conțin date a căror reprezentare este în cod complementar sau în virgulă mobilă, iar înregistrările fizice conțin mai mult de un articol, pot apărea situații în care acestor date să nu le fie asociat zone de memorie cu adrese multiplu de 2, 4 sau 8. Astfel de situații pot conduce la apariția unor erori, la execuția programului, în momentul în care aceste date sunt adresate.

● **Clauza SAME** indică fișierele care vor fi prelucrate:

— în aceleași zone-tampon, dacă sintaxa clauzei este:

**SAME AREA FOR** *nume-fișier-3* [*nume-fișier-4*] . . . ;

— în aceleași zone-tampon și în aceeași zonă-articol, dacă sintaxa clauzei este:

**SAME RECORD AREA FOR** *nume-fișier-3* [*nume-fișier-4*] . . .

### Observații

— Cea de a doua formă de scriere nu poate fi utilizată în cadrul fișierelor prelucrate în mod LOCATE, deoarece nu le sunt rezervate zone-articol.

— Fișierele citate în clauza SAME nu pot fi prelucrate simultan.

— Dimensiunea zonelor-tampon și a zonei-articol rezervate fișierelor citate în clauza SAME se stabilește în funcție de dimensiunea celei mai mari înregistrări fizice și logice ale acestora.

● **Clauza MULTIPLE FILE** indică fișierele membre ale unui multi-fișier pe bandă sau disc magnetic.

*Exemplu.* Secvența următoare:

INPUT-OUTPUT SECTION.  
FILE-CONTROL.

SELECT FISIER-1 ASSIGN TO AMT.

SELECT FISIER-2 ASSIGN TO EMT.

I-O-CONTROL.

MULTIPLE FILE CONTAINS FISIER-1 FISIER-2

APPLY LOCATE MODE ON FISIER-1 FISIER-2.

SAME AREA FOR FISIER-1 FISIER-2.

reprezintă descrierea a două fișiere secvențiale pe bandă magnetică, care alcătuiesc un multifişier. Aceste fișiere vor fi prelucrate în mod LOCATE, în aceleași zone-tampon (prelucrarea în aceleași zone-tampon este posibilă întrucât fișierele membre ale unui multifişier bandă nu pot fi deschise și exploatate simultan).

c. **Rubrica FD.** În secțiunea FILE, fiecărui fișier descris într-o rubrică SELECT, trebuie să-i corespundă o rubrică FD care completează descrierea fișierului din rubrica SELECT. Format general:

**FD** *nume-fișier*

$$\left[ ;\text{RECORDING MODE IS } \begin{Bmatrix} \text{F} \\ \text{V} \\ \text{U} \end{Bmatrix} \right]$$



;LABEL { RECORD IS } { OMITTED  
 RECORDS ARE } { STANDARD }  
 [ ;DATA { RECORD IS }  
 RECORDS ARE } *nume-articol-1* [, *nume-articol-2*] ... ]  
 [ ;RECORD CONTAINS *întreg-1* [TO *întreg-2*] CHARACTERS]  
 [ ;BLOCK CONTAINS *întreg-3* [TO *întreg-4*] { RECORDS  
 CHARACTERS } ].

Clauzele rubricii FD pot fi scrise în orice ordine, iar singura obligatorie este clauza LABEL.

● Clauza **RECORDING** indică formatul articolelor fișierului: *fix* (opțiunea F), *variabil* (opțiunea, V, care este implicită) sau *nedefinit* (opțiunea U).

— Opțiunea F trebuie indicată în cazul fișierelor pe cartele care sînt consultate.

— Opțiunea V poate fi indicată în cazul fișierelor pe cartele (care sînt create), la imprimantă, pe bandă magnetică sau pe disc magnetic.

— Opțiunea U trebuie indicată în cazul consultării fișierelor secvențiale pe bandă sau disc magnetic, ce conțin articole de lungimi diferite, pentru care însă lungimea nu este înregistrată pe suport. Fișierele cu articole de format nedefinit pot fi prelucrate numai în mod LOCATE.

● Clauza **LABEL** indică prezența — prin opțiunea STANDARD — sau absența — prin opțiunea OMITTED — etichetelor de identificare a fișierului. Opțiunea OMITTED poate fi utilizată în descrierea fișierelor al căror suport este cartela, hîrtia de imprimat, banda magnetică, iar opțiunea STANDARD, în descrierea fișierelor al căror suport este banda sau discul magnetic.

● Clauza **DATA** precizează numele articolelor fișierului. Identificatorii *nume-articol-1*, *nume-articol-2* ... reprezintă numele tipurilor de articole care compun fișierul; aceste articole trebuie să fie descrise în continuarea rubricii FD. Clauza DATA este opțională, informația pe care o precizează fiind considerată de compilator comentariu.

● Clauza **RECORD** indică lungimea zonei de memorie necesară memorării articolelor fișierului. În cazul fișierelor cu articole de format fix, lungimea se indică prin *întreg-1*; în cazul articolelor de format variabil sau nedefinit, prin *întreg-1* se indică lungimea minimă a acestei zone, iar prin *întreg-2*, lungimea maximă.

#### Observație

Ca și la clauza DATA, clauza RECORD este opțională, întrucît compilatorul stabilește lungimea zonei necesară memorării articolelor pe baza informațiilor furnizate prin rubricile de descriere a datelor ce compun articolele.

● Clauza **BLOCK** indică numărul de articole conținute de o înregistrare fizică, dacă este utilizată opțiunea RECORDS, sau dimensiunea zonei de memorie (zonei-tampon) necesară memorării înregistrărilor fizice, dacă este utilizată opțiunea CHARACTERS (această opțiune este implicită). Clauza poate fi utilizată numai în descrierea fișierelor pe bandă sau pe disc magnetic.

În cazul fișierelor cu articole de format fix, prin *întreg-3* se indică numărul de articole conținute de o înregistrare fizică sau dimensiunea (în număr de locații) zonei-tampon, după cum este utilizată opțiunea RECORDS și, respectiv, CHARACTERS; în cazul fișierelor cu articole de format variabil,



prin *intreg-3* și *intreg-4* se indică numărul minim și, respectiv, maxim de articole conținute de o înregistrare fizică sau dimensiunea minimă și, respectiv, maximă a zonei-tampon; în cazul fișierelor cu articole de format nedefinit, prin *intreg-3* și *intreg-4* se indică dimensiunea minimă și, respectiv, maximă a zonei-tampon.

#### Observații

— Dacă este utilizată opțiunea RECORDS, dimensiunea zonei-tampon este stabilită de compilator conform tabelii XVIII.1 (unde prin *BFS*, *RCS* *n* și *c* s-au notat: dimensiunea maximă a zonei-tampon, dimensiunea maximă a zonei-articol, numărul de articole conținute de o înregistrare fizică și, respectiv, lungimea cheii de identificare a articolelor).

— Dacă este utilizată opțiunea CHARACTERS, compilatorul verifică dacă în zona-tampon poate fi înregistrat un articol; în caz afirmativ dimensiunea zonei-tampon rămâne cea indicată; în caz contrar, aceasta este stabilită conform relațiilor din tabelul XVIII.1, considerându-se  $n = 1$ .

Tabelul XVIII.1

Mod de organizare	Formatul articolelor	Dimensiunea înregistrărilor fizice (bloc/pagină)
Secvențială	Fix	$BFS = 8 + n(RCS + 1)$
	Variabil	$BFS = 8 + n(RCS + 5)$
Secvențială-indexată	Fix	$BFS = 8 + n(RCS + 6) + K \begin{cases} n = 1, K = 2 \\ n > 1, K = 0 \end{cases}$
	Variabil	$BFS = 8 + n(RCS + 8)$
Selectivă	Fix	$BFS = 8 + n(RCS + c + 2) + K \begin{cases} n = 1, K = 8 \\ n > 1, K = 0 \end{cases}$
	Variabil	$BFS = 8 + n(RCS + c + 4) + K \begin{cases} n = 1, K = 6 \\ n > 1, K = 0 \end{cases}$

### 3. DESCRIEREA OPERAȚIILOR DE ACCES LA FIȘIERE ȘI ARTICOLE

Operațiile de acces la fișiere și la articolele lor sînt descrise în limbajul COBOL prin intermediul instrucțiunilor: OPEN, CLOSE, READ, WRITE și REWRITE.

a. Instrucțiunea OPEN. Descrie operația de *deschidere* a fișierelor, care constă în identificarea și pregătirea lor pentru prelucrare.



Format general:

$$\text{OPEN} \left[ \text{INPUT} \left\{ \text{nume-fișier} \left[ \left\{ \text{REVERSED} \right\} \left\{ \text{WITH NO REWIND} \right\} \right] \dots \right\} \right. \\ \left. \left[ \text{OUTPUT} \left\{ \text{nume-fișier} \left[ \text{WITH NO REWIND} \right] \right\} \dots \right] \right. \\ \left. \left[ \left\{ \text{INPUT-OUTPUT} \right\} \text{nume-fișier} \dots \right] \right. \\ \left. \left[ \text{I-O} \right] \right]$$

Operația de deschidere a unui fișier trebuie să preceadă prima operație de acces (citire sau scriere) la articolele acestuia.

Semnificația opțiunilor este următoarea:

— *INPUT*, *OUTPUT* și *INPUT-OUTPUT/I-O* indică faptul că fișierul va fi consultat, creat și, respectiv, actualizat (pot fi actualizate numai fișierele pe disc magnetic);

— *WITH NO REWIND*, ce poate fi utilizată numai în cazul fișierelor pe bandă magnetică, indică suprimarea rebobinării benzii, care are loc, în mod obișnuit, la deschiderea fișierului;

— *REVERSED*, poate fi utilizată numai în cazul fișierelor pe bandă magnetică, care sînt consultate începînd cu ultimul articol (citire-înapoi).

b. **Instrucțiunea CLOSE.** Descrie operația de *închidere* a fișierelor; în cazul fișierelor pe suporturi reutilizabile această operație constă în crearea sau verificarea etichetelor.

Format general:

$$\text{CLOSE} \left\{ \text{nume-fișier} \left[ \left\{ \text{REEL} \right\} \left[ \left\{ \text{WITH NO REWIND} \right\} \right] \left[ \left\{ \text{WITH LOCK} \right\} \right] \right] \dots \right\}$$

Semnificația opțiunilor este următoarea:

— *REEL* și *UNIT* indică faptul că fișierul ocupă mai multe volume bandă și, respectiv, mai multe volume disc;

— *WITH NO REWIND*, ce poate fi utilizată numai în cazul fișierelor pe bandă magnetică, indică suprimarea rebobinării benzii care are loc, în mod obișnuit la închiderea fișierului;

— *WITH LOCK* indică dezafectarea unității periferice după închiderea unui volum (dacă fișierul ocupă mai multe volume) sau a fișierului.

#### Observații

— Orice fișier care a fost deschis într-un program, trebuie închis înainte încheierii execuției acestuia.

— Un fișier poate fi deschis și închis de mai multe ori în cursul execuției programului în care este definit.

— Opțiunea *WITH NO REWIND*, indicînd poziționarea dispozitivului de scriere/citire al unității de bandă la începutul sau la sfîrșitul unui fișier, poate fi utilizată pentru prelucrarea fișierelor membre ale unui multifîșier, pentru consultarea unui fișier începînd cu ultimul articol, pentru adăugarea de articole la un fișier deja creat etc.

c. **Instrucțiunea WRITE.** Descrie operația de *scriere* (în acces secvențial) a unui articol într-un fișier.

Format general:

$$\text{WRITE } \text{nume-articol} \text{ [FROM } \text{nume-dată}]$$



unde:

- *nume-articol* reprezintă numele unui tip de articol ce aparține fișierului;
  - *nume-dată* indică zona de memorie în care a fost constituit articolul.
- Principiul de execuție al instrucțiunii WRITE este următorul:
- în cazul prelucrării în mod MOVE, articolul este transferat din zona-articol în zona-tampon (în curs de „umplere”), această operație fiind precedată de transferul articolului din zona *nume-dată* în zona-articol, dacă este utilizată opțiunea FROM;
  - în cazul prelucrării în mod LOCATE, articolul este transferat din zona *nume-dată*, în zona-tampon, dacă este utilizată opțiunea FROM;
  - în momentul în care diferența dintre lungimea unei zone-tampon și suma lungimilor articolelor grupate în această zonă este mai mică decât lungimea celui mai mic articol, informația din zona-tampon este înregistrată pe suportul afectat fișierului.

d. **Instrucțiunea READ.** Descrie operația de citire (în acces secvențial) a unui articol dintr-un fișier.

Format general:

**READ** *nume-fișier* [INTO *nume-dată*] AT END *instrucțiune* . . . .

La prima execuție a instrucțiunii READ, o înregistrare fizică a fișierului *nume-fișier* este transferată de pe suportul extern în zona-tampon, iar primul articol al înregistrării fizice este transferat în zona *nume-dată*, dacă este utilizată opțiunea INTO; dacă modul de prelucrare este MOVE acest articol este transferat și în zona-articol. La următoarele execuții ale instrucțiunii READ, următoarele articole ale înregistrării fizice sînt tratate asemănător, fiind citită o altă înregistrare fizică numai în momentul în care toate articolele din zona-tampon au fost prelucrate.

Fiind o instrucțiune condițională, instrucțiunea READ se încheie printr-un punct. Instrucțiunea care urmează lui READ este executată după citirea articolului; la citirea mărcii de sfîrșit de fișier, execuția programului continuă cu prima instrucțiune indicată în clauza AT END.

e. **Instrucțiunea DELETE.** Descrie operația de ștergere a unui articol dintr-un fișier pe disc.

Format general:

**DELETE** *nume-articol*

În vederea actualizării datelor conținute de fișierele pe disc, articolelor li se asociază cîte un *indicator de ștergere*. Inițial (la crearea fișierului), valoarea acestui indicator este 00<sub>16</sub>. Atunci cînd se consideră că anumite articole nu mai sînt necesare, indicatorilor de ștergere asociați li se atribuie valoarea 01<sub>16</sub>, care indică SGF-ului faptul că ele nu mai aparțin „logic” fișierului, fiind neglijate (sărite) la consultarea fișierului. Această operație, numită *invalidare*, reprezintă o *ștergere logică* a articolelor, întrucît ele continuă să ocupe loc pe suport.

Invalidarea se realizează în memorie (în zona-tampon). Din acest motiv, articolele care trebuie șterse sînt mai întîi citite, iar apoi invalidate și rescrise în fișier; citirea se realizează prin instrucțiunea READ, iar invalidarea și rescrierea, prin instrucțiunea DELETE.



f. **Instrucțiunea REWRITE.** Descrie operația de *rescriere* a unui articol citit anterior dintr-un fișier pe disc magnetic, articol ale cărui date au fost modificate total sau parțial.

Format general:

**REWRITE** *nume-articol* [**FROM** *nume-dată*]

Modificarea unui articol se realizează astfel:

- articolul este citit prin intermediul instrucțiunii **READ**;
- uneia sau mai multor date ale articolului li se modifică valorile, prin intermediul uneia sau mai multor instrucțiuni (de exemplu, instrucțiuni de transfer, aritmetice etc.);
- articolul este rescris în fișier, în locul din care a fost citit, prin intermediul instrucțiunii **REWRITE**.

Modificarea valorilor datelor articolului poate fi efectuată în: zona *nume-dată* sau în zona-articol, dacă modul de prelucrare este **MOVE**; zona-tampon, dacă modul de prelucrare este **LOCATE**.

#### Observație

Dacă fișierul conține articole de format variabil, lungimea articolului rescris nu trebuie să depășească lungimea articolului citit anterior din fișier.

## 4. FIȘIERE SECVENȚIALE PE BANDĂ MAGNETICĂ

a. **Caracteristici.** Tipurile de prelucrări la care poate fi supus un fișier pe bandă magnetică sînt crearea și consultarea.

*Crearea* constă în scrierea articolelor pe banda magnetică conform unei anumite ordini (crescătoare sau descrescătoare), dată de o cheie ce identifică articolele, sau fără o ordine stabilită; totodată, scrierea articolelor poate fi precedată și urmată de crearea etichetelor de început și de sfîrșit de fișier. Deoarece banda magnetică nu este un suport adresabil, scrierea înregistrărilor se realizează exclusiv în sensul de parcurgere a benzii.

*Consultarea* constă în citirea articolelor fișierului în vederea unor prelucrări; citirea este precedată de verificarea etichetelor de început de fișier și este urmată de verificarea etichetelor de sfîrșit de fișier. Singurul mod de acces posibil la articolele unui fișier secvențial pe bandă este cel secvențial, care se poate realiza în oricare din cele două sensuri de parcurgere a benzii (citire înainte și citire înapoi).

Articolele fișierelor secvențiale pe bandă magnetică pot fi de format fix, variabil sau nedefinit. Articolelor de format variabil le este asociată, în momentul înregistrării lor pe suport, o informație de patru baiți, primii doi reprezentînd lungimea articolului  $L_a$  (fig. XVIII.1).

Articolele de format fix și variabil pot fi grupate în înregistrări fizice, numite *blocuri*. Blocurile sînt delimitate prin *spații interbloc* (*gap-uri*), și pot conține un antet (fig. XVIII.2) a cărui lungime minimă este de 4 baiți, primii doi baiți reprezentînd numărul de ordine al blocului în cadrul fișierului



Fig. XVIII.1



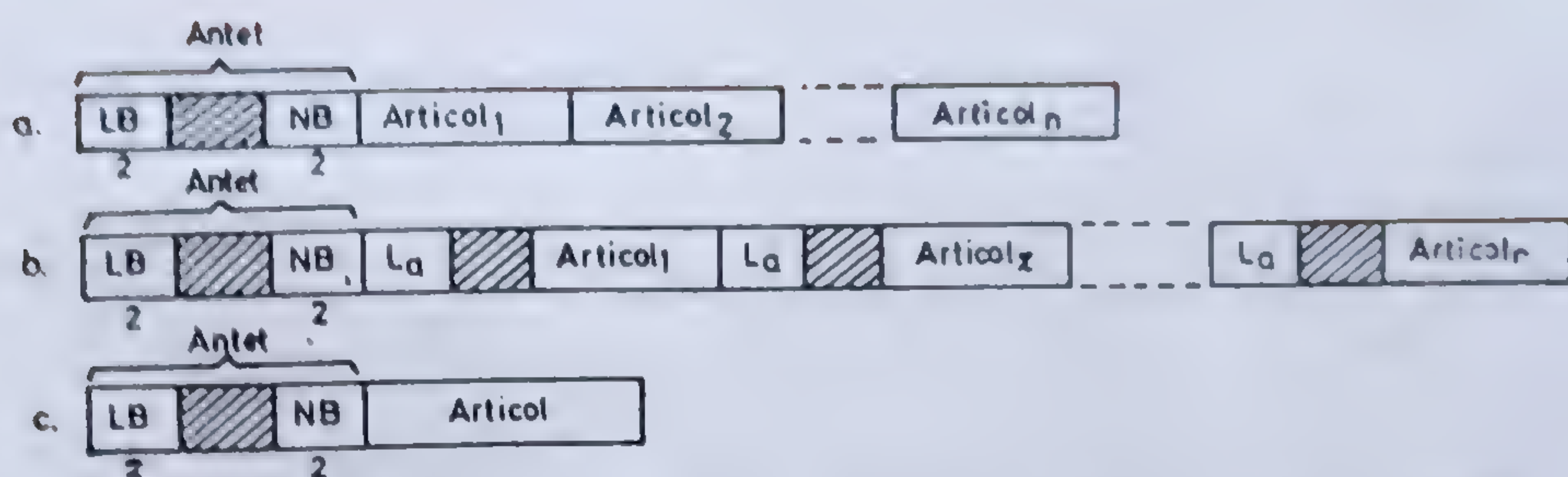


Fig. XVIII.2. a — articole de format fix; b — articole de format variabil; c — articole de format nedefinit.

(NB), iar ultimii doi, lungimea blocului (LB). Prezența antetului trebuie indicată în cartela de comandă FILE, prin argumentul:

MFF:([Inr] [,BLC] [, CBS])

unde:

- *Inr* indică lungimea antetului ( $1 \leq nr \leq 99$ );
- *BLC* indică faptul că antetul conține lungimea blocului;
- *CBS* indică faptul că antetul conține numărul de ordine al blocului în cadrul fișierului.

Lungimea și numărul de ordine al blocului sînt înregistrate în antet la crearea fișierului și sînt verificate la consultarea acestuia. În mod obișnuit blocurile nu au prevăzut antet.

În vederea identificării și protecției, fișierelor pe bandă li se asociază etichete, care sînt grupate în blocuri de lungime 80 de caractere, la începutul și la sfîrșitul fișierelor. În figura XVIII.3. se prezintă dispunerea acestor etichete pe suport în cazul structurilor monovolum — monofișier (fig. XVIII.3a) și monovolum — multifîșier (fig. XVIII.3b). Grupurile de etichete sînt delimitate prin *mărci de fișier* (FM); sfîrșitul unui fișier, sau al unui multifîșier este indicat prin două mărci de fișier.

Semnificația notațiilor din figura XVIII.3 este următoarea:

- VOL1 — eticheta de identificare a volumului;
- HDR1, HDR2 — etichete de început de fișier (conțin informațiile de identificare a fișierului — nume, număr de generație, număr de versiune etc. — și informații referitoare la structura fișierului — formatul articolelor, lungimea articolelor, lungimea blocurilor etc.);
- EOF1, EOF2 — etichete de sfîrșit de fișier (au, în principiu, același conținut ca etichetele HDR1, HDR2);
- EOVI, EOVI — etichete de sfîrșit de volum (au același conținut ca etichetele EOF1, EOF2, fiind create numai dacă sfîrșitul fișierului coincide cu sfîrșitul volumului).

b. **Tipuri de prelucrări.** Crearea și consultarea fișierelor secvențiale pe bandă sînt ilustrate prin intermediul programelor XVIII.1 și XVIII.2. Programul XVIII.1 realizează înregistrarea pe bandă a datelor fișierului pe cartele din programul XV.3. Deoarece asupra acestor date nu se fac prelucrări, ele nu sînt explicitate în descrierea articolului.

În programul XVIII.2 fișierul pe bandă creat prin execuția programului XVIII.1 este consultat, editîndu-se la imprimantă valoarea produselor vin-



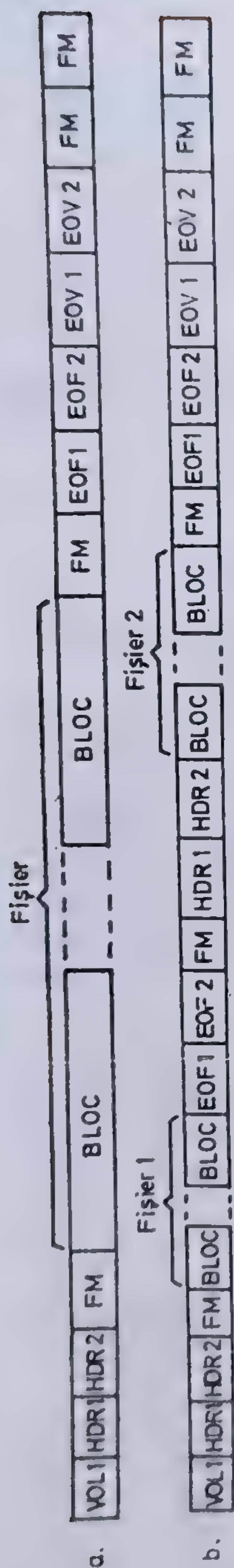


Fig. XVIII.3.

dute în cadrul unui magazin. Codul acestuia este indicat prin intermediul unei cartele parametru. S-a considerat că în fișierul pe bandă nu există o ordonare a articolelor după valorile codurilor magazinelor.

c. **Multifișiere pe bandă magnetică.** În COBOL, un multifișier pe bandă nu face obiectul unei descrieri speciale. În schimb, fișierele membre sînt descrise, ca orice fișiere obișnuite, prin rubricile SELECT și FD, iar numele lor trebuie citate în clauza MULTIPLE FILE. Ordinea în care fișierele membre sînt menționate în clauza MULTIPLE FILE nu are legătură cu ordinea înregistrării acestora pe suport.

Fișierele membre sînt dispuse pe suport secvențial, în ordinea menționată în program; din acest motiv, aceste fișiere nu pot fi deschise și prelucrate în același timp. Utilizarea opțiunii WITH NO REWIND în instrucțiunile OPEN și CLOSE asigură suprimarea rebobinării benzii, care are loc, în mod obișnuit, la deschiderea/închiderea unui fișier membru. În acest fel, se poate trece de la crearea/consultarea unui fișier membru al cărui număr de ordine în cadrul multifișierului este  $n$ , la crearea/consultarea fișierului membru cu numărul de ordine  $n + 1$  (indicînd opțiunea WITH NO REWIND în instrucțiunea CLOSE în care este citat fișierul cu numărul de ordine  $n$  și în instrucțiunea OPEN în care este citat fișierul cu numărul de ordine  $n + 1$ ).

Deoarece SGF-ul nu asigură automat poziționarea dispozitivului de scriere/citire decît la începutul primului fișier membru, pentru a consulta un anumit fișier membru, altul decît primul, trebuie parcurse toate fișierele membre care îl preced pe suport. De asemenea, dacă într-un program sînt prelucrate primele  $m$  fișiere membre ale unui multifișier care conține  $n$  fișiere membre, nu este necesar de a descrie toate cele  $n$  fișiere membre, ci numai primele  $m$ .

#### Observație

Cartelele de comandă care trebuie asociate unui multifișier pe bandă sînt:

— o cartelă FLSSET în care se indică identificatorul de exploatare al multifișierului (identificatorul de exploatare asociat multifișierului coincide cu identificatorul de exploatare al primului fișier citat în clauza MULTIPLE FILE) și tipul multifișierului, prin argumentul MFS MFT;

— o cartelă ASSIGN în care se indică identificatorul de exploatare al multifișierului și adresa simbolică a unității periferice care îi este afectată;

— cîte o cartelă LABEL, pentru fiecare fișier membru, în care se indică informațiile de identificare a acestora.



CREARE

BANDA MAGNETIC

```
JOB COBOL, AN: ISOO, PR: FELIX
COMPILE COBOL
ID DIVISION.
PROGRAM-ID. PVIIII.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT DESFACERI-C ASSIGN TO SYSIN.
    SELECT DESFACERI-B ASSIGN TO ANT.
DATA DIVISION.
FILE SECTION.
FD DESFACERI-C RECORDING F LABEL RECORD OMITTED.
01 PRODUS-C PIC X(61).
FD DESFACERI-B RECORDING F LABEL RECORD STANDARD
    BLOCK CONTAINS 15 RECORDS.
01 PRODUS-B PIC X(61).
WORKING-STORAGE SECTION.
01 SF PIC 9 VALUE ZERO.
PROCEDURE DIVISION.
CREARE.
    OPEN INPUT DESFACERI-C OUTPUT DESFACERI-B
    PERFORM CITIRE-SCRIERE UNTIL SF = 1.
    CLOSE DESFACERI-C DESFACERI-B
    STOP RUN.
CITIRE-SCRIERE.
    READ DESFACERI-C AT END MOVE 1 TO SF.
    IF SF = 0
        THEN
            WRITE PRODUS-B FROM PRODUS-C.
LINK
INIT DVT:NT,VS:NTIMAN
ASSIGN A,DVT:NT,VS:NTIMAN
LABEL A,FN: DESFACERI
RUN
```

```
*
* FISIER DESFACERI.C
*
```

END

COBOL

PROGRAMUL XVIII.1

## 5. FIȘIERE SECVENȚIALE PE DISC MAGNETIC

a. Caracteristici. Fișierele secvențiale pe disc magnetic admit trei tipuri de prelucrări: creare, consultare și actualizare.

Crearea se realizează, în principiu, la fel ca în cazul fișierelor pe bandă, ordinea în care se scriu articolele fiind stabilită de utilizator.

Consultarea se poate realiza numai în acces secvențial și numai începând cu primul articol.



# CONSULTARE BANDA MAGN.

```

JOB COROL,AN:IS00,PN:FELIX
COMPILE COBOL
ID DIVISION.
PROGRAM-ID. FVIII2.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT DESFACERI ASSIGN TO AMT.
DATA DIVISION.
FILE SECTION.
FD DESFACERI RECORDING F LABEL RECORD STANDARD
    BLOCK CONTAINS 15 RECORDS.

01 PRODUS.
    05 COD-MAGAZIN PIC 99.
    05 FILLER PIC X(47).
    05 VALOARE PIC 9(10)V99.
WORKING-STORAGE SECTION.
01 SF PIC 9 VALUE ZERO.
01 PARAMETRU.
    05 COD-M PIC 99.
    05 DEN-M PIC X(20).
01 VAL-TOTALA PIC 9(12)V99 VALUE ZERO
01 VAL-TOTALA-E PIC Z(11)9.99.
PROCEDURE DIVISION.
CONSULTARE.
    OPEN INPUT DESFACERI
    ACCEPT PARAMETRU
    READ DESFACERI AT END MOVE 1 TO SF.
    PERFORM TEST UNTIL SF = 1.
    MOVE VAL-TOTALA TO VAL-TOTALA-E.
    DISPLAY ' MAGAZIN: ' DEN-M ' VALOARE: ' VAL-TOTALA-E
    CLOSE DESFACERI
    STOP RUN.

TEST.
    IF COD-MAGAZIN = COD-M
        THEN
            ADD VALOARE TO VAL-TOTALA.
    READ DESFACERI AT END MOVE 1 TO SF.
LINK
ASSIGN A,DVT:MT,VS:MT1MAN
LABEL A,FN:'DESFACERI'
RUN
EOJ

```

## PROGRAMUL XVIII.2

*Actualizarea*, care implică consultarea fișierului, constă în ștergeri sau modificări de articole.

Pe disc pot fi organizate două tipuri de fișiere secvențiale:

— fișiere secvențiale *continue* (numite, în mod curent, fișiere secvențiale), în cazul în care articolele sînt înregistrate succesiv în zone distincte ale suportului, rezervate numai acestor fișiere;

— fișiere secvențiale *înlănțuite*, în cazul în care articolele sînt înregistrate pe suport, succesiv sau nu, în zone rezervate mai multor fișiere (care alcătuiesc multifiișiere pe disc), numite *zone partajate*.

Articolele pot fi de format fix sau variabil. Fiecărui articol, i se poate asocia, la crearea fișierului, un indicator de ștergere IS. Prezența indicatorului de ștergere trebuie cerută în cartela de comandă FILE, prin argumentul MFF: (DLC). Articolelor de format variabil li se asociază, la crearea fișierului, 4 baiți, primii doi reprezentînd lungimea articolului  $L_0$  (fig. XVIII.4). Atît articolele de format fix cît și cele de format variabil pot fi grupate în *blocuri*. Deoarece scrierea/citirea pe/de pe disc se realizează de la începutul



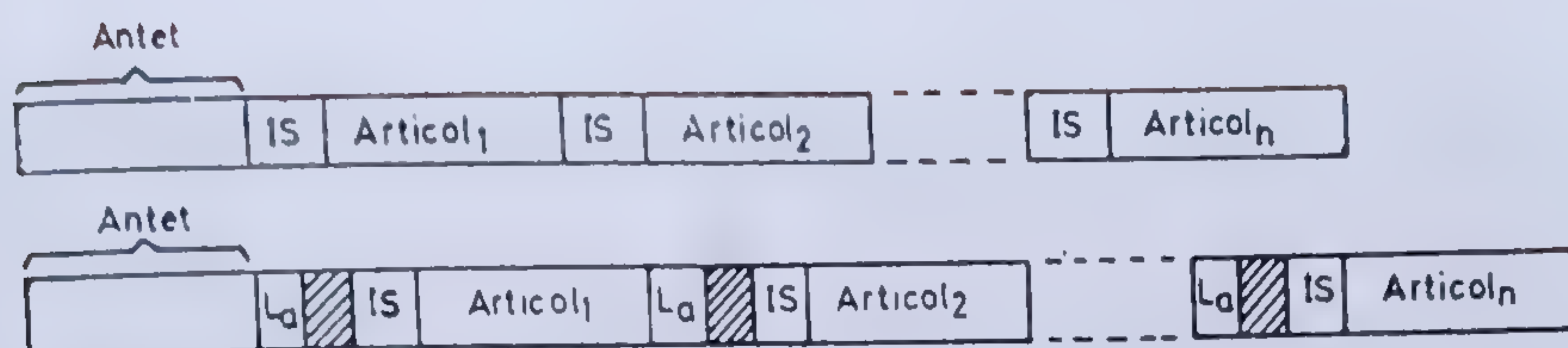


Fig. XVIII.4. a — articole de format fix; b — articole de format variabil.

unui sector, blocurile ocupă un număr întreg de sectoare (care trebuie să fie divizor al numărului de sectoare care alcătuiesc un cilindru). Partea din bloc care conține efectiv informații se numește *pagină*; lungimea unei pagini este cel mult egală cu lungimea blocului. Fiecărei pagini i se asociază 8 baiți (antet) care reprezintă informații de control, înregistrate și utilizate de SGF pentru exploatarea fișierului. În cazul fișierelor secvențiale înlanțuite, întrucât articolele pot fi înregistrate, în pagini diferite ale zonei partajate (SGF-ul alocând dinamic aceste pagini fișierelor ce compun multifîșierul), paginile sînt înlanțuite prin adrese.

Identificarea și protecția fișierelor pe disc magnetic se realizează printr-un sistem standard de etichete grupate în primul cilindru al volumului (zona de control al volumului). Fiecărui fișier aflat pe volum îi sînt asociate: o etichetă HDR1, care conține informațiile de identificare a fișierului, și o etichetă HDR2, care conține informații privind structura fișierului. Multifîșierelor disc li se asociază cîte o etichetă HDR1 și HDR2 și, pentru fiecare fișier membru, cîte o etichetă HDR3 (care conține informațiile de identificare a acestora).

b. **Tipuri de prelucrări.** Pentru a ilustra prelucrarea fișierelor secvențiale pe disc magnetic se consideră *exemplele* prezentate în continuare.

• Fie un fișier pe cartele RETETE-CARTELE care conține informații privind rețetele de fabricație a produselor unei întreprinderi industriale. Întreprinderea are trei secții, în cadrul cărora se fabrică produse distincte. În alcătuirea unui produs pot intra minimum trei materiale și maximum opt materiale. În fișierul RETETE-CARTELE, pentru fiecare produs există un grup de minimum patru și maximum 9 articole. Primul articol din grup conține datele de identificare a produsului: codul secției (1 caracter numeric), codul produsului (6 caractere numerice) și denumirea produsului (20 caractere alfanumerice). Următoarele articole din grup conțin fiecare date referitoare la un material care intră în alcătuirea produsului: codul produsului (6 caractere numerice), codul materialului (8 caractere numerice), denumirea materialului (20 caractere alfanumerice), procentul de material care concurează la obținerea unei unități din produsul respectiv (2 caractere numerice).

Considerînd că valorile datelor fișierului RETETE-CARTELE sînt corecte, se cere ca, din articolele acestui fișier, să se creeze un fișier secvențial pe disc RETETE-DISC, cu articole de format variabil. Articolele fișierului pe disc vor grupa, fiecare, datele referitoare la un anumit produs și vor avea: o parte fixă (care se regăsește în toate articolele) ce va conține codul secției, codul produsului, denumirea produsului și numărul de materiale care concurează la obținerea produsului; o parte variabilă, ce va conține, pentru fiecare material, codul, denumirea și procentul (fig. XVIII.5).

COD-SECȚIE-D	COD-PRODUS-D	DENUMIRE-PRODUS-D	MATERIAL-D	PARTE-VARIABILĂ		
				COD-MATERIAL-D	DENUMIRE-MATERIAL-D	PROCENT-D
9	916	X120	9	918	X120	99

Fig. XVIII.5



Rezolvarea este prezentată în programul XVIII.3.

● În programul XVIII.4 este prezentată operația de actualizare a fișierului RETETE, creat în programul XVIII.3. Informațiile necesare actualizării sînt furnizate de fișierul pe cartele TRANZACTII, care conține două tipuri de articole. Un articol de primul tip conține două date: TIP-ACT (1 caracter alfabetic), a cărei valoare este S și COD-PRODUS-T (6 caractere numerice); acest articol indică ștergerea, din fișierul RETETE, a articolului al cărui cod produs coincide cu cel de pe cartelă. Un articol de tipul doi conține patru date: TIP-ACT (1 caracter alfabetic), a cărei valoare este M; COD-PRODUS-T (6 caractere numerice); COD-MATERIAL-T (8 caractere numerice); PROCENT-T (2 caractere numerice). Acest articol indică modificarea, în fișierul RETETE, a articolului al cărui cod produs coincide cu cel de pe cartelă; modificarea constă în atribuirea valorii datei PROCENT-T datei PROCENT ce corespunde materialului al cărui cod coincide cu cel de pe cartelă.

În rezolvarea problemei s-a considerat că articolele celor două fișiere sînt ordonate crescător după codul produsului. De asemenea, s-a considerat că pot exista mai multe modificări relative la același produs (fig. XVIII.6).

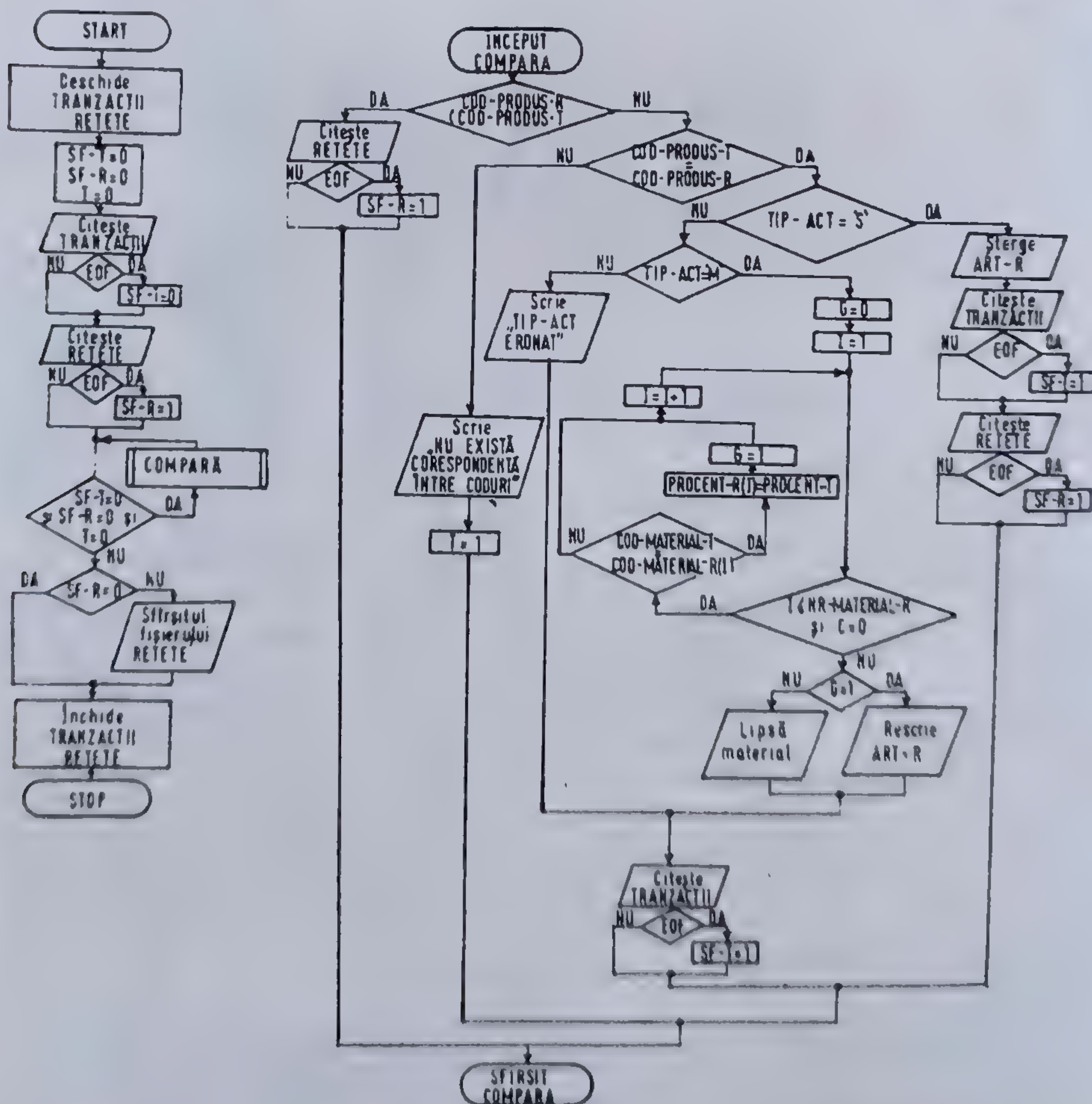


Fig. XVIII.6



c. **Multifișiere pe disc magnetic.** Ca și în cazul unui multifișier pe bandă un multifișier pe disc nu face obiectul unei descrieri aparte, fiind descrise numai fișierele membre. Fiecărui fișier membru trebuie să-i corespundă o rubrică SELECT și o rubrică FD. De asemenea, numele fișierelor membre trebuie citate în clauza MULTIPLE FILE.

Spre deosebire de fișierele membre ale unui multifișier pe bandă, fișierele membre ale unui multifișier pe disc pot fi deschise și prelucrate în același timp, deoarece suportul este adresabil. De asemenea, în cazul în care într-un program care consultă *n* fișiere membre ale unui multifișier disc, nu trebuie descrise decât acestea.

#### Observație

Informațiile care trebuie furnizate prin cartelele de comandă, în cazul prelucrării multifișierelor pe disc, sînt:

- identificatorul de exploatare al multifișierului (care este stabilit după aceeași regulă ca și în cazul unui multifișier bandă) și tipul multifișierului (prin argumentul MFS:MFA) — în cartela FLSET;

- identificatorul de exploatare complet al fișierului membru (format dintr-o literă, urmată de o cifră care indică tipul prelucrării: 1 — consultare, 2 — creare, 3 — actualizare), tipul unității periferice și tipul multifișierului (prin argumentul MFS:MFA) de care aparține fișierul membru — în cartela FILE (pentru fiecare fișier membru trebuie furnizată o cartelă FILE);

- identificatorul de exploatare al multifișierului și adresa simbolică a unității periferice afectată acestuia — în cartela de comandă ASSIGN;

- identificatorul de exploatare al fișierului membru și numele acestuia (prin argumentul FN: 'nume-multifișier/nume-fișier-membru') — în cartela LABEL (pentru fiecare fișier membru trebuie furnizată o cartelă LABEL).

Dacă multifișierul este creat, trebuie furnizată și o cartelă de comandă ALLOC, în care se indică numărul de cilindri alocați zonei-partajate.

## 6. FIȘIERE SECVENȚIALE-INDEXATE

a. **Caracteristici.** Un fișier este organizat secvențial-indexat dacă:

- articolele care îl compun, care pot fi de format fix sau variabil, sînt ordonate strict crescător după valorile unei date ce reprezintă *cheia de identificare* a articolelor;

- suportul (care trebuie să fie adresabil) este împărțit, în vederea înregistrării și regăsirii articolelor, în trei zone numite: *partea principală* — în care se înregistrează articolele la crearea fișierului, *tabela de indecși* — care realizează corespondența între valorile cheii de identificare și unitățile adresabile (pagină, cilindru, volum) în care acestea sînt înregistrate și *partea de depășire* — în care se înregistrează articolele adăugate ulterior fișierului.

Determinarea poziției unui articol *x* în fișier se poate realiza în două moduri:

- în *acces secvențial*, prin parcurgerea tuturor articolelor a căror valoare a cheii este strict mai mică decât cea a articolului *x*;

- în *acces direct*, prin consultarea tabelelor de indecși pentru a stabili pagina care conține articolul *x*, iar în cadrul paginii, prin parcurgerea tuturor articolelor a căror valoare a cheii este mai mică decât cea a articolului *x*.



Tipurile de prelucrări ce se pot efectua asupra fișierelor secvențial-indexate sînt, în funcție de modul de acces, următoarele:

- *crearea*, în acces secvențial;
- *consultarea*, în acces secvențial sau direct;
- *actualizarea*, care constă în modificarea sau ștergerea unor articole, în cazul accesului secvențial, și modificarea, ștergerea sau adăugarea unor articole, în cazul accesului direct.

b. **Structură.** Structura părții principale a unui fișier secvențial-indexat este prezentată în figura XVIII.7.

În partea principală articolele sînt grupate în pagini cărora le este asociat cîte un antet de 8 baiți, structurat astfel:

- I (1 bait) care indică tipul paginii (de sfîrșit de fișier, de sfîrșit de zonă etc.);
- A (5 baiți), care indică adresa relativă disc a articolului cu valoarea cheii cea mai mică din pagină;
- LP (2 baiți), care indică lungimea paginii.

Fiecare articol este precedat de un indicator de ștergere IS și este urmat de o adresă de înlănțuire AI (adresa relativă a articolului care urmează în secvența definită de valorile cheii de identificare). În cazul în care articolele sînt de format variabil, lungimea fiecărui articol LA (2 baiți) urmează indicatorului de ștergere.

**Tabelele de indecși** se constituie la crearea fișierului și nu se modifică în timpul prelucrării ulterioare a acestuia.

Fiecărui tip de unitate adresabilă (pagină, cilindru, volum) ocupată de fișier, îi pot fi asociate două tipuri de tabele:

- *tabela detaliată*, care conține, pentru fiecare tip de unitate adresabilă, adresa acesteia și valoarea cheii ultimului articol înregistrat în unitatea adresabilă;
- *tabela rezumat*, care conține, pentru fiecare pagină ocupată de tabela detaliată, adresa acesteia și valoarea cea mai mare a cheii din pagina respectivă.

Adresa paginii, cilindrului sau a volumului și valoarea cheii constituie un *index*. Indecșii sînt grupați în pagini a căror lungime se stabilește în funcție de lungimea cheii conform tabelului XVIII.2, unde prin LC s-a notat lungimea cheii.

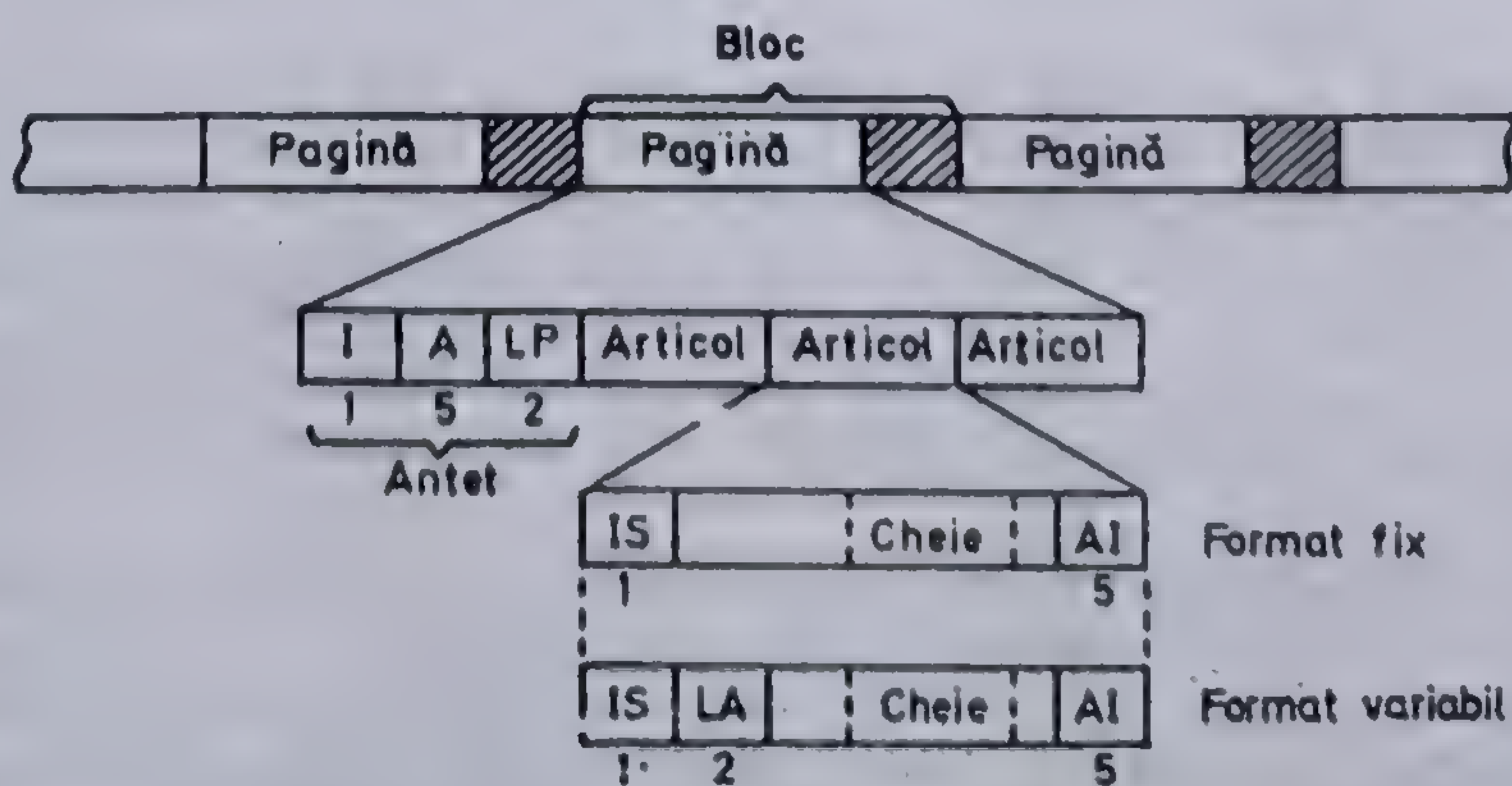


Fig. XVIII.7



TABLIUL XVIII.2

Lungimea cheii	Dimensiunea paginii
$1 \leq LC \leq 13$ baiți	256 baiți
$14 \leq LC \leq 30$ baiți	512 baiți
$31 \leq LC \leq 47$ baiți	768 baiți
$48 \leq LC \leq 64$ baiți	1024 baiți

Structura unei table de indcși este prezentată în figura XVIII.8.

Partea de depășire reprezintă zona alocată fișierului în vederea înregistrării articolelor ce se adaugă după crearea fișierului.

Structura părții de depășire este asemănătoare celei prezentate în figura XVIII.7; deosebirea constă în faptul că informația A, din antetul paginii, nu este utilizată, iar informația LA este asociată și articolelor de format fix. Lungimea paginilor este egală cu cea a paginilor din partea principală.

c. Clauze de descriere. În funcție de modul de acces utilizat, în descrierea unui fișier secvențial-indexat trebuie să se indice obligatoriu:

— modul de organizare și cheia de identificare a articolelor, în cazul accesului secvențial;

— modul de organizare, modul de acces, cheia de identificare a articolelor — prin clauza RECORD KEY — și cheia de acces — prin clauza SYMBOLIC KEY — în cazul accesului direct.

● Clauza RECORD, al cărei format general este:

**RECORD KEY IS nume-dată**

indică data (grupată sau elementară) din cadrul articolelor fișierului, care reprezintă cheia de identificare. În cazul în care fișierul este constituit din mai multe tipuri de articole, data *nume-dată* trebuie să aibă aceeași lungime și poziție în cadrul tuturor tipurilor de articole.

● Clauza SYMBOLIC, al cărei format general este:

**SYMBOLIC KEY IS nume-dată**

indică data (grupată sau elementară) care constituie cheia de acces la articole.

#### Observație

Nu poate fi definită drept cheie de identificare sau cheie de acces o dată descrisă cu clauza OCCURS sau subordonată unei date descrise cu această clauză.

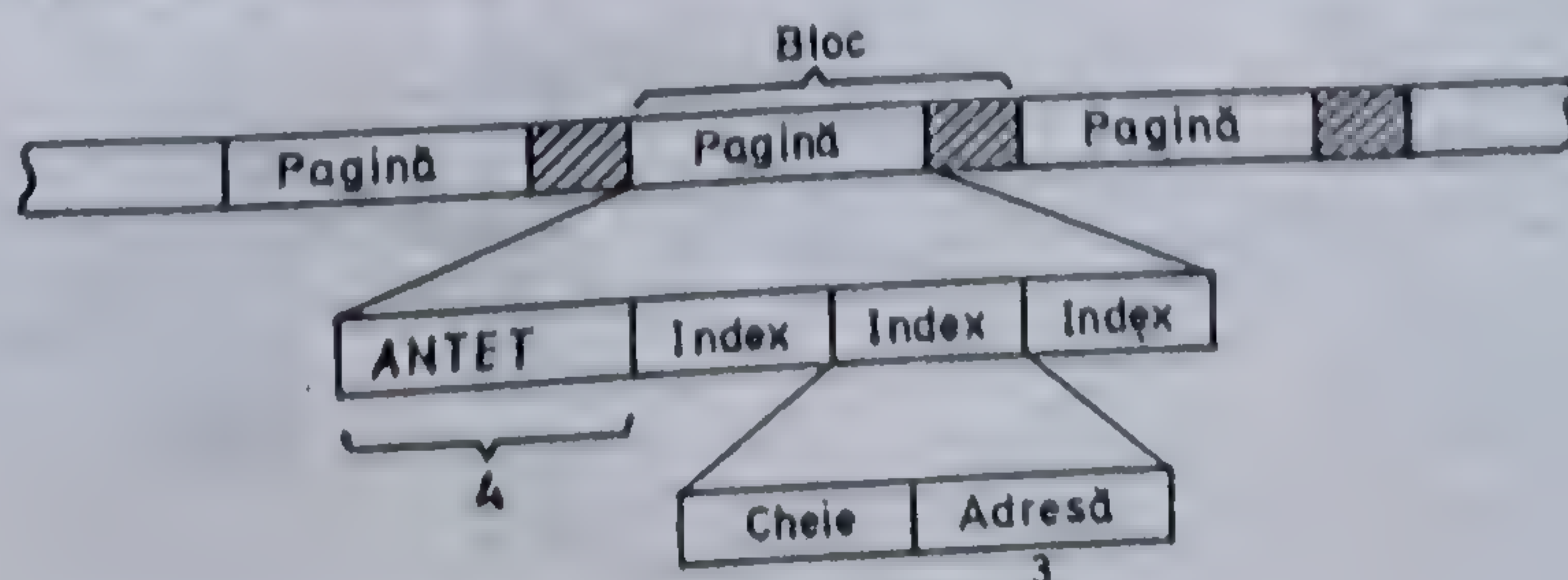


Fig. XVIII.8



d. Crearea unui fișier secvențial-indexat constă în înregistrarea pe suport, în zona alocată părții principale, a articolelor ce compun fișierul. Schematic, un fișier secvențial-indexat se poate reprezenta ca în figura XVIII.9, unde, numerele: 1, 7, 12, 14, 17, 20, 21, 23, 24, 27, 30 și 35 reprezintă valorile cheii de identificare a celor 12 articole ce compun fișierul. După crearea fișierului, informația A din antetul paginilor are valoarea zero, deoarece articolele cu cea mai mică valoare a cheii din fiecare pagină, aparțin părții principale. De asemenea, deoarece toate articolele fișierului se află în partea principală, adresa de înlănțuire asociată fiecărui articol, are valoarea zero.

Operația de scriere a unui articol într-un fișier secvențial-indexat se indică prin instrucțiunea WRITE pentru acces direct:

**WRITE** *nume-articol* [FROM *nume-dată*]

**INVALID KEY** *instrucțiune*.

Clauza INVALID KEY descrie operația de comparare a valorii  $v_i$  a cheii articolului ce urmează a fi scris cu valoarea  $v_{i-1}$  a cheii articolului scris anterior. Dacă  $v_i > v_{i-1}$  articolul este scris, iar execuția programului continuă cu execuția instrucțiunii ce urmează după punct; în caz contrar, execuția programului continuă cu instrucțiunea (instrucțiunile) indicată în clauza INVALID KEY.

*Exemplu.* În programul XVIII.5 este descrisă operația de creare a fișierului secvențial-indexat BIBLIOTECA, din articolele fișierului pe cartele FIȘIER-CARTI.

Articolele fișierului pe cartele conțin date referitoare la cărțile existente într-o bibliotecă și anume: codul cărții, titlul cărții, autorul și numărul de exemplare și sînt ordonate crescător după valorile datei COD-CARTE-C.

Cheia de identificare a articolelor este data COD-CARTE-B.

e. Consultarea fișierelor secvențiale-indexate se poate realiza atît în acces secvențial cît și în acces direct.

Consultarea în acces secvențial constă în transferul succesiv al articolelor din fișier în memorie, începînd cu primul articol. Această operație este descrisă de instrucțiunea READ pentru acces secvențial.

Consultarea în acces direct constă în citirea numai a anumitor articole, a căror poziție în fișier este determinată prin intermediul tabelelor de indecși.

Pentru a citi un articol în acces direct, în zona asociată datei definită SYMBOLIC KEY trebuie să se înregistreze valoarea cheii acestuia. De exemplu, pentru consultarea în acces direct a fișierului prezentat în figura XVIII.9, în vederea citirii articolului a cărui cheie are valoarea 23, se realizează următoarele operații:

— atribuirea valorii 23 datei definite SYMBOLIC KEY;

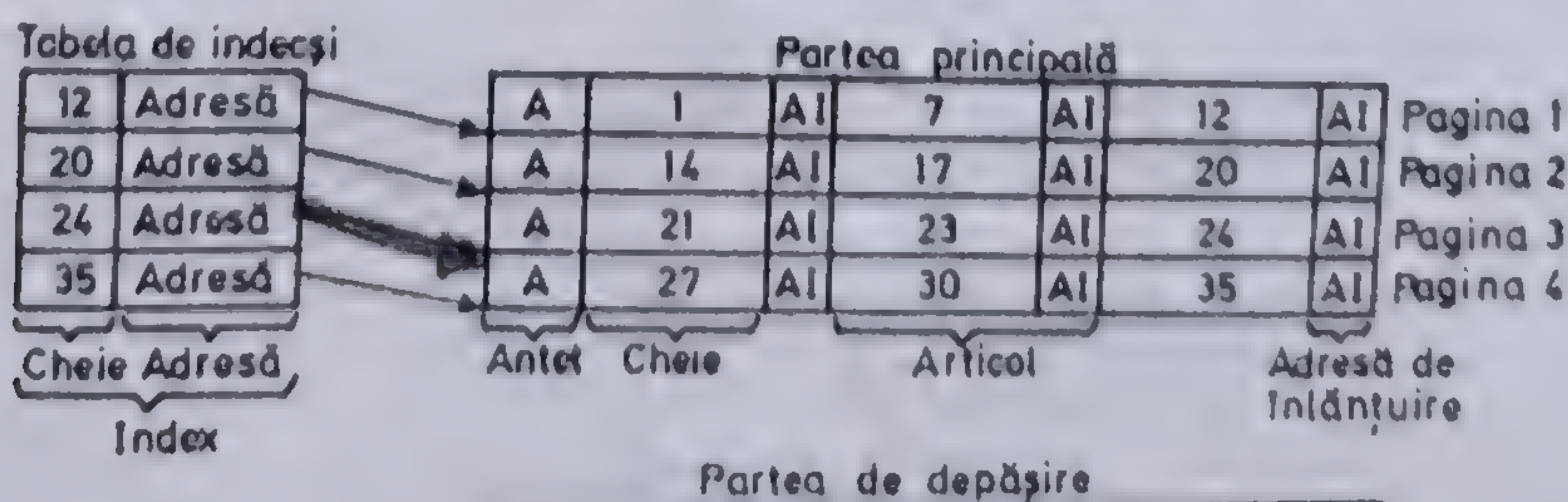


Fig. XVIII.9



# CREATE SECN - INDEXATA

JOB COROL, AN: IS00, FN: FELIX  
COMPILE COROL  
ID DIVISION.  
PROGRAM-ID. PVIIIS.  
ENVIRONMENT DIVISION.  
CONFIGURATION SECTION.  
INPUT-OUTPUT SECTION.  
FILE-CONTROL.

RECORD KEY - close  
identification  
etc.

SELECT FISIER-CARTI ASSIGN TO SYSIN.  
SELECT BIBLIOTECA ASSIGN TO AAD ORGANIZATION INDEXED  
ACCESS SEQUENTIAL RECORD KEY IS COD-CARTE-B.

DATA DIVISION.  
FILE SECTION.

FD FISIER-CARTI RECORDING F LABEL RECORD OMITTED.

01 ART-C.

05 COD-CARTE-C PIC 9(8).  
05 TITLU-C PIC X(10).  
05 AUTOR-C PIC A(10).  
05 NR-EXEMPLARE-C PIC 9(5).

FD BIBLIOTECA RECORDING F LABEL RECORD STANDARD  
BLOCK CONTAINS 20 RECORDS.

01 ART-B.

05 COD-CARTE-B PIC 9(8) COMP-3.  
05 TITLU-B PIC X(10).  
05 AUTOR-B PIC A(10).  
05 NR-EXEMPLARE-B PIC 9(5) COMP-3.

securizat împachetat  
 $(\frac{N}{2} + 1)$

WORKING-STORAGE SECTION.

01 SF PIC 9 VALUE 0.  
01 EROARE-CHEIE PIC 99.  
NR-ART PIC 9(6) VALUE 0.

PROCEDURE DIVISION.

PROGRAM-PRINCIPAL.

OPEN INPUT FISIER-CARTI OUTPUT BIBLIOTECA  
READ FISIER-CARTI AT END MOVE 1 TO SF.  
PERFORM SCRIE UNTIL SF = 1.  
DISPLAY 'IN FISIERUL BIBLIOTECA S-AU SCRIS: ' NR-ART ' ART  
CLOSE FISIER-CARTI BIBLIOTECA  
STOP RUN.

SCRIE.

MOVE COD-CARTE-C TO COD-CARTE-B  
MOVE TITLU-C TO TITLU-B  
MOVE AUTOR-C TO AUTOR-B  
MOVE NR-EXEMPLARE-C TO NR-EXEMPLARE-B  
MOVE 0 TO EROARE-CHEIE  
WRITE ART-B INVALID KEY MOVE 1 TO EROARE-CHEIE.  
IF EROARE-CHEIE = 1

THEN  
DISPLAY '\*\*\* CHEIE ERONATA \*\*\*' ART-C  
ELSE

ADD 1 TO NR-ART.  
READ FISIER AT END MOVE 1 TO SF.

LINK  
ASSIGN A, DVIIAD, VSTADIMAN  
LABEL A, FN, BIBLIOTECA, ANIANY, SZIL  
RUN

\* FISIER 'FISIER-CARTI'

.EOF

COJ

PROGRAMUL NVIIIS



— determinarea indexului, din tabela detaliată, a cărui cheie are valoarea mai mare decât 23;

— căutarea, în pagina a cărei adresă este specificată de indexul determinat anterior, a articolului pentru care valoarea cheii este 23.

#### Observație

În cazul în care fișierul conține tabele de index organizate la mai multe nivele, căutarea unui articol începe cu consultarea tabelii rezumat (detaliată) de nivel maxim.

Operația de citire în acces direct se indică prin instrucțiunea **READ** pentru acces direct:

**READ** nume-fișier [**INTO** nume-dată] **INVALID KEY** instrucțiune.

În cazul în care articolul este găsit în fișier, execuția programului continuă cu instrucțiunea ce urmează instrucțiunii **READ**; în caz contrar, execuția continuă cu instrucțiunea (instrucțiunile) indicată în clauza **INVALID KEY**.

*Exemplu.* Programul XVIII.6 realizează consultarea în acces direct a fișierului secvențial-indexat **BIBLIOTECA**, creat prin programul XVIII.5. Valorile cheilor de identificare a articolelor ce sînt citite sînt furnizate de articolele fișierului pe cartele **FISIER-CHEI**. Deoarece valorile cheii de identificare a articolelor sînt reprezentate în zecimal-împachetat, cheia de acces a fost definită data **COD-CARTE-W**, descrisă în secțiunea **WORKING**.

Pentru fiecare articol citit din fișierul **BIBLIOTECA**, se înregistrează la imprimantă un rînd care conține codul cărții, titlul și autorul.

f. **Actualizarea fișierelor secvențiale-indexate.** În cadrul acestei prelucrări se pot realiza, funcție de modul de acces, următoarele operații:

- modificarea și/sau ștergerea unor articole din fișier, în acces secvențial;
- modificarea, ștergerea și/sau adăugarea unor articole din/în fișier,

în acces direct.

Indiferent de modul de acces, operațiile de modificare și ștergere sînt precedate de o operație de consultare a fișierului, prin care, articolul ce urmează a fi modificat sau șters este introdus în memorie. Această operație este descrisă de instrucțiunea:

**READ** nume-fișier [**INTO** nume-dată] **AT END** instrucțiune.

dacă actualizarea se realizează în acces secvențial, și de instrucțiunea:

**READ** nume-fișier [**INTO** nume-dată] **INVALID KEY** instrucțiune.

dacă actualizarea se realizează în acces direct.

● Operația de rescriere în fișier a unui articol ce a fost modificat se indică prin instrucțiunea **REWRITE**:

**REWRITE** nume-articol [**FROM** nume-dată]

**INVALID KEY** instrucțiune.

Instrucțiunea (instrucțiunile) citată în clauza **INVALID KEY** va fi executată în cazul în care, prin modificarea valorilor datelor articolului citit anterior din fișier, valoarea cheii de identificare este schimbată.

● Operația de ștergere a unui articol este indicată prin instrucțiunea **DELETE**:

**DELETE** nume articol



DISCUȚIE ARII INDEX ARII INDEX

```

JOB COBOL,AN:IS00,PN:FELIX
COMPILE COBOL
TO DIVISION.
PROGRAM-ID. PVIII.6.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT BIBLIOTECA ASSIGN TO AAO ORGANIZATION INDEXED
        ACCESS RANDOM RECORD COD-CARTE-B SYNOCLIC CODCARTE-W.
    SELECT FISIER-CHEI ASSIGN TO SYSIN.
DATA DIVISION.
FILE SECTION.
FD BIBLIOTECA,RECORDING F LABEL RECORD STANDARD
    BLOCK 20 RECORDS.
01 ART-B.
    05 COD-CARTE-B          PIC 9(8) COMP-3.
    05 TITLU-B              PIC X(10).
    05 AUTOR-B              PIC A(10).
    05 NR-EXEMPLARE-B       PIC 9(5) COMP-3.
FD FISIER-CHEI RECORDING F LABEL RECORD OMITTED.
01 ART-C.
    05 COD-CARTE-C          PIC 9(8).
WORKING-STORAGE SECTION.
01 SF                      PIC 9 VALUE 0.
01 IVK                      PIC 9.
01 COD-CARTE-W              PIC 9(8) COMP-3.
PROCEDURE DIVISION.
PROGRAM-PRINCIPAL..
    OPEN INPUT FISIER-CHEI BIBLIOTECA
    READ FISIER-CHEI AT END MOVE 1 TO SF.
    PERFORM PRELUCRARE UNTIL SF = 1
    CLOSE FISIER-CHEI BIBLIOTECA
    STOP RUN.
PRELUCRARE.
    MOVE COD-CARTE-C TO COD-CARTE-W
    MOVE 0 TO IVK
    READ BIBLOTECA INVALID KEY MOVE 1 TO IVK.
    IF IVK = 0
        THEN
            DISPLAY ART-B
        ELSE
            DISPLAY 'ARTICOLUL CU CHEIA ' COD-CARTE-C ' NU EXISTA'.
    READ FISIER-CHEI AT END MOVE 1 TO SF.
    LINK
    ASSIGN A,DUT:AD,US:ADIMAN
    LABEL A,PN:'BIBLIOTECA'
    RUN
*
* FISIER 'FISIER-CHEI'
*

```

COBOL

PROGRAMUL XVIII.6

În cazul fișierelor secvențiale-indexate, operația de ștergere se realizează prin invalidare pentru a nu se distruge înlanțuirea articolelor.

● Adăugarea de noi articole în fișierele secvențiale-indexate, operație indicată prin instrucțiunea WRITE pentru acces direct, se realizează astfel încât tabelele de indecși nu sînt modificate. De exemplu, articolul pentru care valoarea cheii este 25 va fi adăugat fișierului prezentat în figura XVIII.9, în partea de depășire, dar va fi considerat articolul cu cea mai mică valoare a cheii din pagina 4. Pentru aceasta, în zona ce corespunde informației A



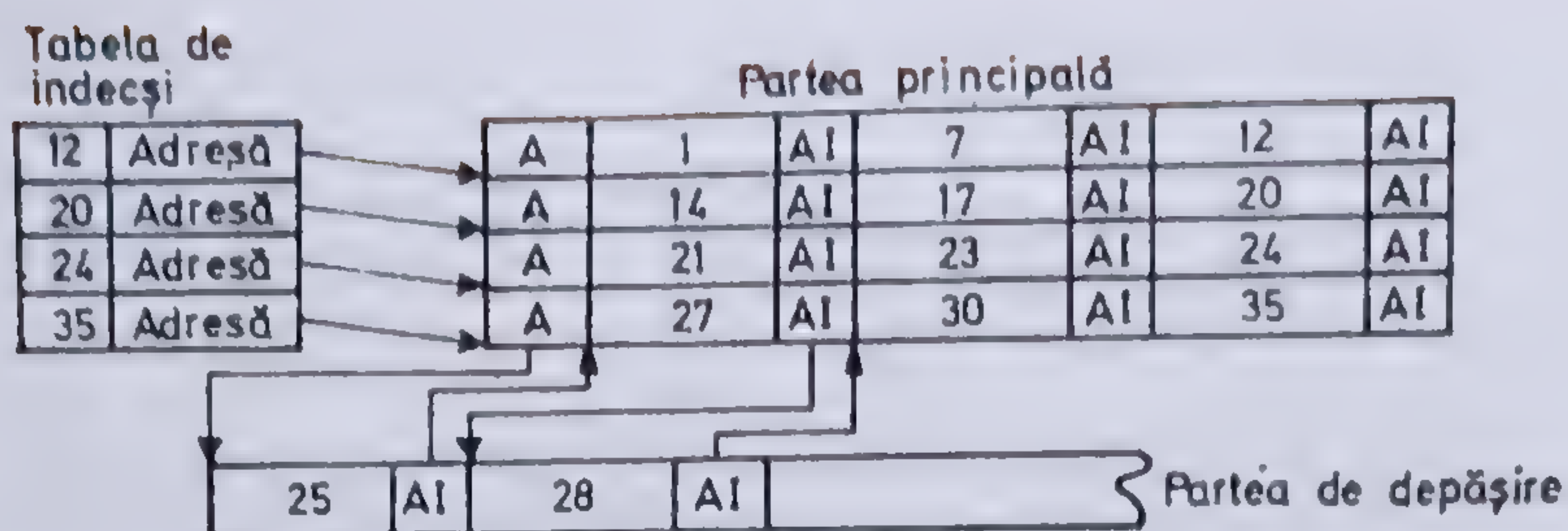


Fig. XVIII.10

din antetul paginii 4 va fi înregistrată adresa relativă a acestui articol în partea de depășire ( $A \neq 0$ ), iar în zona ce corespunde informației AI a articolului adăugat, va fi înregistrată adresa relativă a articolului pentru care valoarea cheii este 27. Prin adăugarea articolului pentru care valoarea cheii este 28, în zona ce corespunde informației AI a articolului pentru care valoarea cheii este 27, este înregistrată adresa articolului adăugat în partea de depășire, iar în zona ce corespunde informației AI a articolului adăugat se înregistrează adresa relativă a articolului pentru care valoarea cheii este 30. După efectuarea operațiilor descrise, fișierul va avea conținutul prezentat în figura XVIII.10.

Înainte de a se adăuga în fișier un articol, valoarea cheii de identificare a acestuia trebuie atribuită datei declarată SYMBOLIC KEY. În cazul în care se cere adăugarea în fișier a unui articol pentru care valoarea cheii există deja la un articol al fișierului, va fi executată instrucțiunea (instrucțiunile) citată în clauza INVALID KEY a instrucțiunii WRITE.

*Exemplu.* Programul XVIII.7 realizează actualizarea, în acces direct, a fișierului BIBLIOTECA, prin intermediul fișierului pe cartele TRANZACȚII. Articolele fișierului TRANZACȚII conțin următoarele date:

- TIP-ACT, care indică prin valorile A, S și M, tipul operației de actualizare: adăugare, ștergere, modificare;
- COD-CARTE-C, care indică valorile cheilor de identificare a articolelor din fișierul BIBLIOTECA ce trebuie adăugate, șterse sau modificate;
- TITLU-C, AUTOR-C și NR-EXEMPLARE-C.

În cazul adăugării, articolul ART-C conține toate datele enumerate, în cazul ștergerii, numai datele TIP-ACT și COD-CARTE-C, iar în cazul modificării, datele TIP-ACT, COD-CARTE-C și NR-EXEMPLARE (valoarea acestei date reprezintă numărul de exemplare cu care este mărit stocul cărții cu codul COD-CARTE-C).

## 7. FIȘIERE SELECTIVE

a. **Caracteristici.** Un fișier este organizat selectiv dacă:

— articolele care îl compun sînt grupate în  $n$  clase distincte, conform unei funcții ce asociază fiecărui articol clasa din care acesta face parte; articolele dintr-o clasă se numesc *sinonime*, iar algoritmul după care se calculează valorile funcției, *algoritm de randomizare*;

— articolele sinonime pot fi identificate în mod unic prin *cheia de identificare*;

— suportul (care trebuie să fie adresabil) este împărțit, în vederea înregistrării articolelor, în două zone numite: *partea principală* — formată din  $n$



zone de aceeași lungime numite *căsuțe* (fiecărei căsuță îi este asociată o clasă) — și *partea de depășire*.

Articolele din fiecare clasă sînt scrise în căsuța care îi corespunde; dacă într-o căsuță nu pot fi scrise toate articolele din clasa respectivă, scrierea se continuă în partea de depășire, legătura cu căsuța asociată realizîndu-se prin adrese.

b. **Structură.** În figura XVIII.11 este prezentată structura părții principale. O pagină din partea principală este afectată uneia sau mai multor căsuțe; fiecărei căsuțe îi este asociat un antet de 7 baiți, structurat astfel:

- A (5 baiți), care indică adresa primului articol (din clasa ce corespunde căsuței) înregistrat în partea de depășire;
- LC (2 baiți), care indică lungimea căsuței.

La înregistrarea articolelor în căsuțe, SGF-ul asociază fiecărui articol valoarea cheii de identificare a acestuia și, eventual, un contur de actualizare CA (2 baiți), căruia îi atribuie valoarea zero. La consultarea fișierului, valoarea contorului de actualizare va fi mărită cu o unitate după fiecare citire a articolului, indicînd, la un moment dat, numărul de citiri ale articolului.

În figura XVIII.12 este prezentată structura părții de depășire. O pagină din partea de depășire (a cărei lungime este egală cu cea a paginii din partea principală) conține un antet de 7 baiți și unul sau mai multe articole.

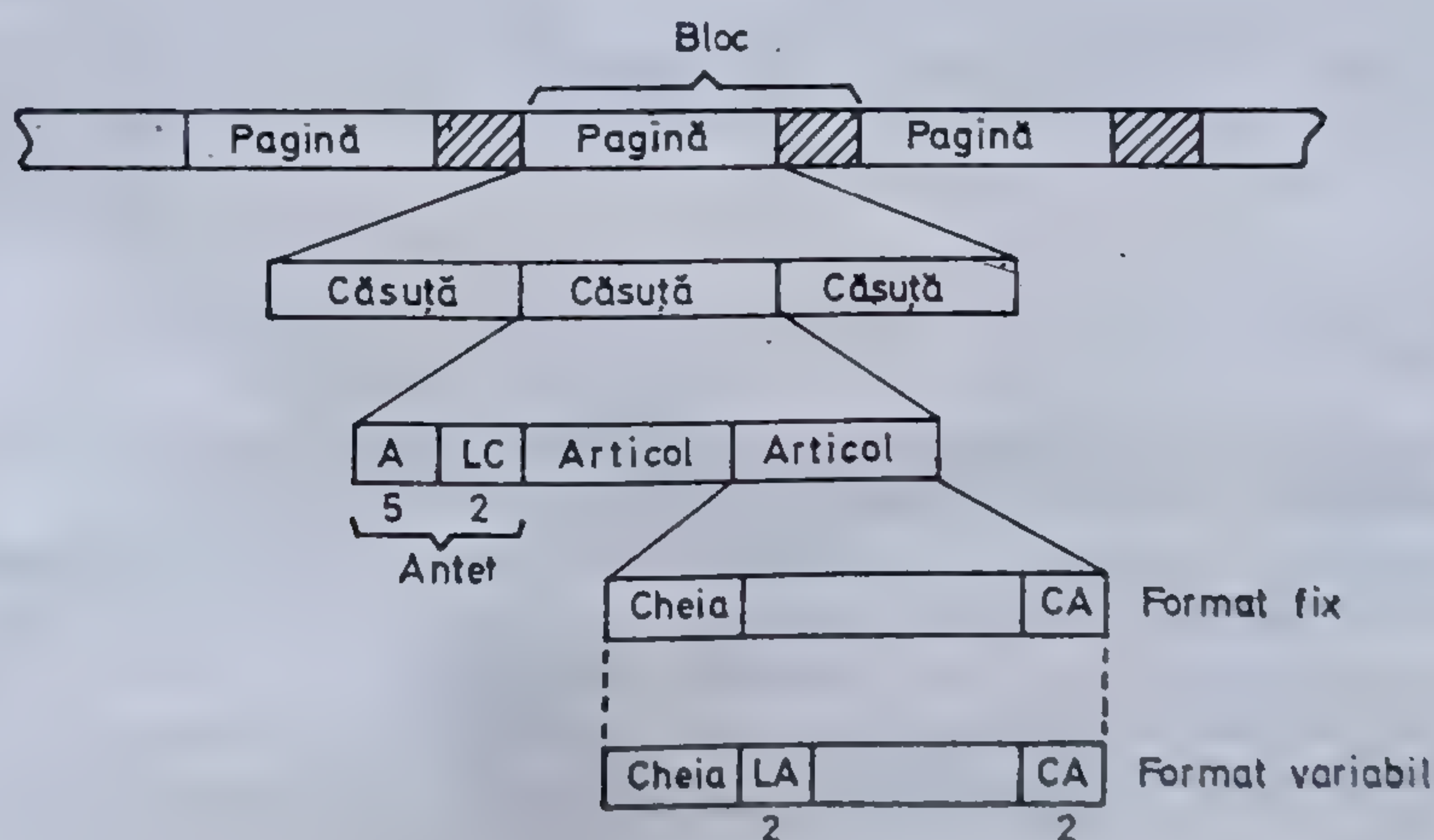


Fig. XVIII.11

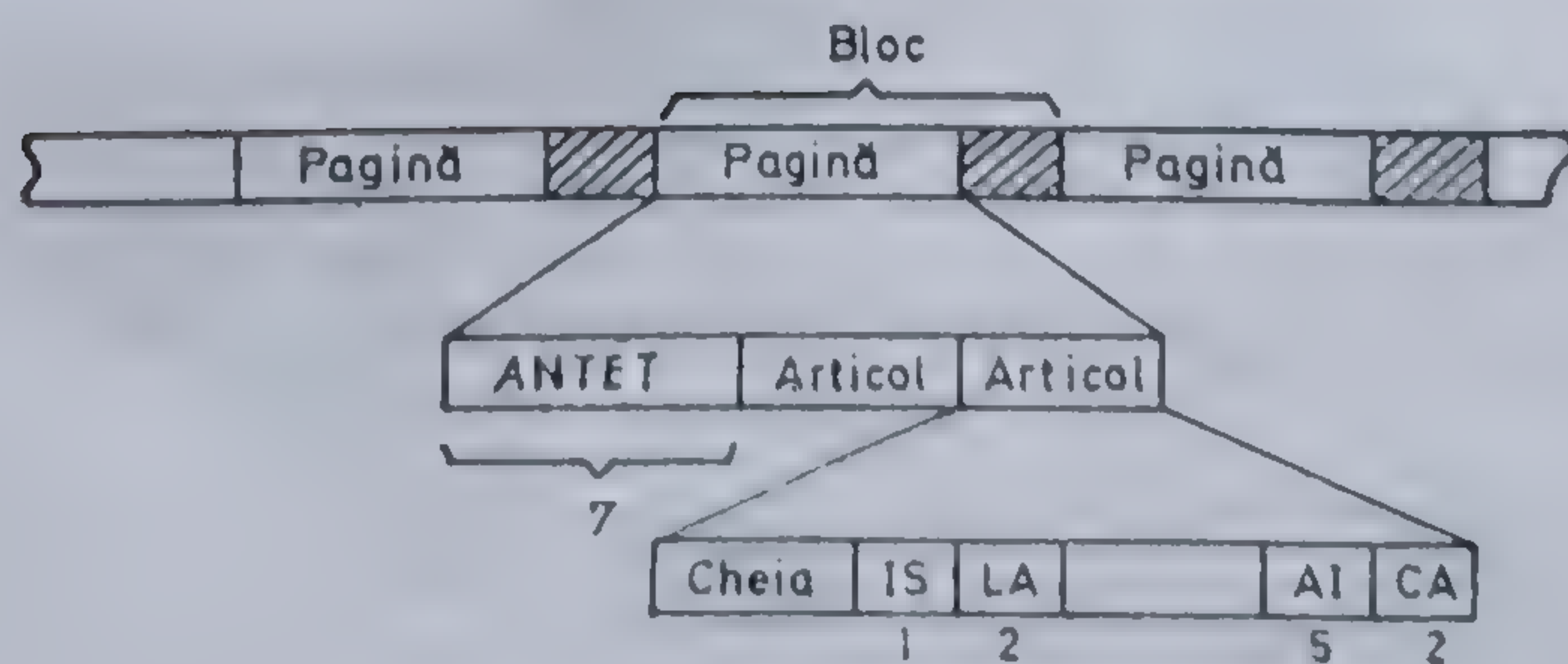


Fig. XVIII.12



Fiecărui articol înregistrat în partea de depășire SGF-ul îi asociază următoarele informații:

- valoarea cheii de identificare;
- indicatorul de ștergere IS (1 bait);
- lungimea LA (2 baiți);
- adresa de înlănțuire (adresa articolului sinonim următor) AI (5 baiți);
- controlul de actualizare CA (2 baiți).

c. **Algoritmul de randomizare** reprezintă un șir de operații efectuate, de obicei, asupra valorii cheii de identificare a articolelor. Rezultatul obținut este un număr cuprins între 0 și  $n - 1$ , care reprezintă numărul căsuței asociate unei clase de articole; pe baza acestui număr, SGF-ul determină adresa fizică a căsuței.

*Exemplu.* Se consideră un fișier alcătuit din 6 articole a căror structură este următoarea:

COD	DENUMIRE	ALTE-INFORMAȚII
-----	----------	-----------------

Data COD are, corespunzător celor 6 articole, valorile: 6, 3, 12, 5, 18, 23.

Pentru a organiza selectiv acest fișier se poate stabili, de exemplu, drept cheie de identificare a articolelor, data COD. Un algoritm de randomizare simplu, prin care se realizează gruparea celor 6 articole în 3 clase, constă în împărțirea valorilor cheilor de identificare la 3; restul obținut reprezintă numărul căsuței. Prin aplicarea acestui algoritm, componența celor trei clase de articole va fi următoarea:

clasa 1 : 6   3   12   18  
 clasa 2 :  
 clasa 3 : 5   23

Considerând că o căsuță poate conține maximum 3 articole, distribuția articolelor în fișier este prezentată în figura XVIII.13.

Algoritmul prezentat nu realizează o distribuție uniformă a articolelor în cadrul căsuțelor; în căsuța 1 nu a fost înregistrat nici-un articol, iar în căsuța 0, nemaifiind loc pentru înregistrarea articolului pentru care valoarea cheii este 18, acesta a fost înregistrat în partea de depășire.

O distribuție mai bună poate fi realizată prin algoritmul de randomizare care constă în împărțirea valorilor cheii la 2 (figura XVIII.14).

6	3	12	Căsuța 0
			Căsuța 1
5	23		Căsuța 2

} Partea principală

18	
----	--

} Partea de depășire

Fig. XVIII.13

6	12	18	Căsuța 0
3	5	23	Căsuța 1

} Partea principală

Fig. XVIII.14



Stabilirea unui algoritm de randomizare care să realizeze o distribuție uniformă a articolelor în căsuțe este, în general, dificilă, deoarece valorile cheii de identificare nu au o distribuție uniformă.

Din acest motiv, organizarea selectivă este mai puțin utilizată în practică, deși este mult mai performantă din punctul de vedere al tipului de acces la articole.

d. **Clauze de descriere specifice.** Indiferent de tipul de prelucrare la care este supus un fișier selectiv, în rubrica SELECT trebuie să se indice obligatoriu:

— modul de organizare, prin clauza ORGANIZATION MODE IS DIRECT;

— modul de acces, prin clauza ACCESS MODE IS RANDOM;

— cheia de identificare, prin clauza SYMBOLIC KEY, și cheia de acces, prin clauza ACTUAL KEY.

● **Clauza SYMBOLIC KEY** indică data căreia trebuie să i se atribuie, înainte de scrierea/citirea unui articol într-o/dintr-o anumită căsuță, valoarea cheii de identificare a articolului. În cazul în care fișierul este creat, SGF-ul asociază articolului această valoare, în scopul identificării ulterioare. În cazul în care fișierul este consultat, SGF-ul utilizează această valoare pentru căutarea articolului în cadrul căsuței.

● **Clauza ACTUAL KEY** indică data căreia trebuie să i se atribuie, înainte de scrierea/citirea unui articol, numărul căsuței asociată clasei din care acesta face parte. Acest număr trebuie să fie reprezentat în cod complementar într-un cuvânt-memorie. Deci, data declarată ACTUAL KEY trebuie descrisă astfel:

PIC S9(9) COMP.

În clauza BLOCK se indică lungimea paginii sau numărul de articole din căsuță.

e. **Tipuri de prelucrări și instrucțiuni specifice.** Fișierele selective pot fi create, consultate sau actualizate. Aceste prelucrări se realizează numai în acces direct și trebuie să fie precedate de operația de constituire a valorilor a cheilor (care constă în atribuirea de valori datelor declarate SYMBOLIC KEY și ACTUAL KEY).

● **Crearea fișierelor selective.** Operația de scriere a unui articol, este descrisă de instrucțiunea WRITE pentru acces direct:

WRITE nume-articol [FROM nume-dată]  
INVALID KEY instrucțiune.

Articolul este înregistrat în căsuța al cărei număr coincide cu valoarea datei declarată ACTUAL KEY, numai dacă acest număr este cuprins între 0 și  $n - 1$  ( $n$  reprezintă numărul total de căsuțe din fișier) și în această căsuță nu există un articol pentru care valoarea cheii să coincidă cu cea a cheii articolului ce trebuie scris; în caz contrar se execută instrucțiunea (instrucțiunile) indicată prin clauza INVALID KEY.

*Exemplu.* Se consideră un fișier pe cartele FURNIZORI-CARTELE care conține informații referitoare la furnizorii unei întreprinderi. Fiecare articol ART-C grupează informații despre un furnizor: cod, denumire, adresă, cont de decontare la bancă.



Din articolele acestui fișier să se creeze un fișier selectiv FURNIZORI ale cărui articole vor conține: denumirea, adresa și contul de decontare ale unui furnizor. Cheia de identificare a articolelor în cadrul căsuțelor va fi codul furnizorului. Fișierul FURNIZORI va conține 50 de căsuțe, fiecare căsuță va ocupa o pagină, iar într-o pagină vor fi scrise 9 articole.

Considerînd că valorile datei COD-FURNIZOR au o distribuție uniformă, numărul căsuței în care e scris un anumit articol va reprezenta cîtul împărțirii valorii datei COD-FURNIZOR la 20.

Rezolvarea este prezentată în programul XVIII.8.

Deoarece SGF-ul asociază fiecărui articol al fișierului FURNIZORI cheia de identificare, codul furnizorului nu figurează în descrierea articolului

```

      JOB COBOL,AN:IS00,PN:FELIX
      COMPILE COBOL
      ID DIVISION.
      PROGRAM-ID. FVIII.8.
      ENVIRONMENT DIVISION.
      CONFIGURATION SECTION.
      INPUT-OUTPUT SECTION.
      FILE-CONTROL.
          SELECT FURNIZORI-CARTELE ASSIGN TO SYSIN.
          SELECT FURNIZORI ASSIGN TO AAD ORGANIZATION DIRECT
              ACCESS RANDOM SYMBOLIC COD-FURNIZOR ACTUAL NR-CASUTA.
      DATA DIVISION.
      FILE SECTION.
      FD FURNIZORI-CARTELE RECORDING F LABEL RECORD OMITTED.
      01 ART-C.
          05 COD-FURNIZOR          PIC 999.
          05 DENUMIRE-FURNIZOR     PIC X(20).
          05 ADRESA-FURNIZOR       PIC X(20).
          05 CONT-FURNIZOR         PIC 9(8).
      FD FURNIZORI RECORDING F LABEL RECORD STANDARD
          BLOCK 9 RECORDS.
      01 ART-F.
          05 DENUMIRE-FURNIZOR     PIC X(20).
          05 ADRESA-FURNIZOR       PIC X(20).
          05 CONT-FURNIZOR         PIC 9(8).
      WORKING-STORAGE SECTION.
      01 NR-CASUTA                PIC S9(5) COMP.
      01 SF                      PIC 9 VALUE 0.
      PROCEDURE DIVISION.
      START.
          OPEN INPUT FURNIZORI.
          OPEN OUTPUT FURNIZORI.
          READ FURNIZORI-CARTELE AT END MOVE 1 TO SF.
          PERFORM SCRIE-CITESTE UNTIL SF = 1.
          CLOSE FURNIZORI-CARTELE FURNIZORI.
          STOP RUN.
      SCRIE-CITESTE.
          MOVE CORRESPONDING ART-C TO ART-F.
          COMPUTE NR-CASUTA = COD-FURNIZOR / 20.
          WRITE ART-F INVALID KEY DISPLAY 'CHEIE ERONATA' COD-FURNIZOR.
          READ FURNIZOR-CARTELE AT END MOVE 1 TO SF.
          FILE A2,DVT:AD,FBN:50,FBN:1
          LINK
          INIT DVT:AD,VS:AD:MAN
          ASSIGN A,DVT:AD,VS:AD:MAN
          LABEL A,FBN:'FURNIZORI',AM:ANY,SZ:9
          RUN
      *
      * FISIER FURNIZORI-CARTELE
      *
      .EOF
      EOJ
      PROGRAMUL XVIII.8
  
```



ART-F. Numărul total de căsuțe și numărul de căsuțe conținute de o pagină au fost precizate în cartela FILE, prin argumentele FBN și respectiv, PBN.

● **Consultarea fișierelor selective.** Pentru a citi un articol dintr-un fișier selectiv trebuie să se cunoască valoarea cheii de identificare a articolului și numărului căsuței în care acesta se află. Aceste valori trebuie atribuite, înainte de citirea articolului, datelor declarate SYMBOLIC KEY și, respectiv, ACTUAL KEY.

Operația de citire este descrisă de instrucțiunea READ pentru acces direct:

**READ** *nume-fișier* [**INTO** *nume-dată*] **INVALID KEY** *instrucțiune*.

Instrucțiunea (instrucțiunile) indicată prin clauza **INVALID KEY** va fi executată dacă numărul căsuței înregistrat în zona asociată datei declarată **ACTUAL KEY** nu este cuprinsă între 0 și  $n - 1$ , sau dacă în această căsuță nu există articolul căutat.

● **Actualizarea fișierelor selective.** Prin actualizare se pot realiza modificări, ștergeri și/sau adăugări de articole.

Operațiile de modificare sau de ștergere a unui articol trebuie să fie precedate de citirea articolului respectiv, care se realizează prin intermediul instrucțiunii **READ** pentru acces direct.

Instrucțiunile prin care se realizează adăugarea, ștergerea și rescrierea articolelor modificate sînt:

**WRITE** *nume-articol* [**FROM** *nume-dată*] **INVALID KEY** *instrucțiune*.

**DELETE** *nume-articol*

**REWRITE** *nume-articol* [**FROM** *nume-dată*]

În cazul fișierelor selective, ștergerea unui articol este fizică, dacă articolul se află în partea principală (în locul lui putînd fi înregistrat un nou articol), sau logică, dacă articolul se află în partea de depășire.

## PROBLEME

1. Să se actualizeze fișierul pe bandă creat în programul XVIII.1 prin intermediul unui fișier pe cartele ale cărui articole conțin aceleași date ca și articolele fișierului pe bandă (cod magazin, denumire magazin, cod produs, denumire produs, unitate de măsură, cantitate, preț unitar, valoare); în plus, articolele fișierului pe cartele conțin în coloana 80 un indicator ale cărui valori — 1, 2 sau 3 — indică adăugarea, ștergerea și modificarea de articole din fișierul pe bandă. Prin actualizare va rezulta un nou fișier pe bandă. Să considerăm că fișierul de actualizat și fișierul pe cartele au articolele ordonate crescător după valorile datelor cod magazin și cod produs.
2. Să se consulte fișierul pe bandă creat la punctul 1 și să se editeze un raport care să conțină pentru fiecare magazin un rînd cu datele următoare: cod magazin, valoarea produselor vîndute. Titlul raportului va fi scris la începutul unei pagini, iar pe fiecare pagină vor fi scrise 50 de rînduri.
3. Din articolele fișierului pe bandă creat la punctul 1 să se creeze un multifîșier pe bandă cu două fișiere membre. În primul fișier membru vor fi scrise articolele care au codul maga-



zinului mai mic sau egal decât 100, iar în cel de al doilea, articolele care au codul magazinului mai mare decât 100.

4. Să se afișeze la imprimantă articolele celui de-al doilea fișier membru ce aparține multi-fișierului pe bandă creat la punctul 3.
5. Din articolele fișierului RETETE, creat în programul XVIII.4, să se creeze un fișier secvențial-indexat; cheia de identificare a articolelor va fi codul produsului.
6. Să se actualizeze, în acces secvențial, fișierul secvențial-indexat creat la punctul 5, prin intermediul unui fișier pe cartele ale cărui articole să indice operații de modificare și ștergere (operația de modificare va consta în schimbarea rețetelor de fabricație ale anumitor produse).
7. Să se creeze din articolele fișierului indexat secvențial BIBLIOTECA (vezi programul XVIII.5) un fișier selectiv. Cheia de identificare a articolelor în cadrul căsuțelor va fi codul cărții. Numărul căsuțelor în care vor fi scrise articolele se va stabili astfel:
  - căsuța 0 va conține articolele pentru care valoarea cheii de identificare aparține intervalului (0; 9999999);
  - căsuța 1 va conține articolele pentru care valoarea cheii de identificare aparține intervalului (10000000; 19999999) ș.a.m.d.
8. Să se afișeze la imprimantă articolele fișierului selectiv creat la punctul 7.



## CAPITOLUL XIX

### PRELUCRĂRI SPECIALE ÎN COBOL

Scopul acestui capitol este de a prezenta unele din prelucrările speciale ce pot fi realizate în limbajul COBOL. Pentru două dintre acestea — utilizarea subprogramelor și segmentarea programelor — o prezentare de principiu s-a realizat la limbajul FORTRAN; pentru sortarea fișierelor și editarea de rapoarte, facilități nedisponibile în FORTRAN, există în cadrul acestui capitol prezentări complete.

#### 1. SUBPROGRAME COBOL

Apelarea unui subprogram se realizează prin intermediul instrucțiunii CALL, al cărui *format* este următorul:

**CALL** *nume* [**USING** *nume-dată-1* [*nume-dată-2*] . . .]

unde *nume* reprezintă numele subprogramului iar clauza **USING** indică parametrii transmiși subprogramului; *nume-dată-1*, *nume-dată-2*, . . . desemnează date descrise în programul apelant, în rubrici de descriere ale căror numere de nivel sînt 01 sau 77.

În cazul în care nu este necesară transmiterea de parametri subprogramului clauza **USING** poate lipsi.

Comparativ cu un program COBOL obișnuit, un subprogram COBOL prezintă următoarele *particularități*:

- parametrii transmiși subprogramului sînt specificați în clauza **USING** din titlul diviziunii **PROCEDURE**, în aceeași ordine în care au fost citați în clauza **USING** a instrucțiunii **CALL**, prin același nume sau prin nume diferite;

- descrierea parametrilor se realizează în secțiunea **LINKAGE**; caracteristicile lor (categoria, lungimile, formatele de reprezentare, alinierea) trebuie să corespundă celor indicate în descrierile parametrilor din programul apelant;

- revenirea din subprogram este asigurată de instrucțiunea **EXIT PROGRAM**, care trebuie să fie singura instrucțiune a unui paragraf.

Numele subprogramului este indicat în paragraful **PROGRAM-ID**; acest nume trebuie să fie identic cu cel indicat în instrucțiunea **CALL** prin care subprogramul este apelat în vederea execuției.

#### Observație

- Un subprogram COBOL nu trebuie să conțină instrucțiuni **CALL** care să apeleze programul apelant, dar poate conține instrucțiuni de apel al altor subprograme.

*Exemplu.* Se consideră un fișier pe cartele **CARTELE** ale cărui articole conțin informații referitoare la cărțile unei biblioteci. Primele 25 de caractere ale fiecărui articol reprezintă titlul cărții, iar următoarele 55, diverse informații. Din articolele acestui fișier va fi creat un fișier selectiv **CARTI** ale



cărui articole vor avea aceeași structură ca articolele fișierului CARTELE și vor fi înregistrate în 26 de căsuțe astfel: în prima căsuță vor fi înregistrate toate articolele care conțin titluri de cărți ce încep cu litera A, în a doua căsuță vor fi înregistrate toate articolele care conțin titluri de cărți ce încep cu litera B ș.a.m.d. Cheia de identificare a articolelor (SYMBOLIC KEY) în cadrul căsuțelor este data care reprezintă titlul cărții.

Să se scrie programul COBOL pentru crearea fișierului CARTI astfel încât determinarea numerelor căsuțelor în care vor fi înregistrate articolele să se realizeze în cadrul unui subprogram COBOL. Rezolvarea problemei enunțate este prezentată în programul XIX.1.

## 2. SEGMENTAREA PROGRAMELOR COBOL

Două metode pot fi utilizate pentru segmentarea programelor COBOL.

a. **Prima metodă**, care permite obținerea unei structuri arborescente cu mai multe nivele, necesită structurarea programului în unități de program distincte (program principal, subprograme) și indicarea modului de înlănțuire a segmentelor ce corespund acestor unități de program.

Alcătuirea și compoziția segmentelor în cazul programului COBOL este oarecum particulară, deoarece acestea nu sînt precizate prin cartele SEG, ci sînt stabilite de compilator.

Astfel, prin compilarea unei unități de program rezultă un modul de program, un modul de date și unul sau mai multe module de date comune (egal cu numărul fișierelor care nu utilizează unitățile standard ale sistemului).

Modulul de program conține instrucțiunile în limbaj-mașină ce corespund instrucțiunilor COBOL. Modulul de date conține datele definite în diviziunea DATA precum și datele incluse implicit de compilator în programul obiect (constantele necesare execuției programului, tabelele de descriere a fișierelor, literalele specificate în diviziunea PROCEDURE, rezultatele intermediare ale unor operații etc.).

Modulul de date comune conține zonele-tampon asociate unui anumit fișier.

Numele modulelor generate sînt stabilite prin adăugarea la primele șase caractere ale numelui programului (indicat în paragraful PROGRAM-ID) a:

- caracterelor 01, în cazul modulului de program;
- caracterelor 00, în cazul modulului de date;
- identificatorul de exploatare al fișierului (o literă urmată de una din cifrele 1, 2, 3 sau 4 — care indică tipul prelucrării acestuia), în cazul modulului de date comune (dacă numele programului este format din mai puțin de șase caractere, se completează pînă la șase cu caracterul %).

Compilatorul COBOL asociază modulelor generate o cartelă SEG, care indică numele ce va fi atribuit, în faza de editare a legăturilor, segmentelor de program și de date; acest nume coincide cu numele modulului de date generat.

Din textele modulelor obiect rezultate, editorul de legături va genera: un segment de program, care conține modulul de program, subprograme din BSS (Biblioteca de Subprograme Standard), module de acces SGF etc.; un segment de date, care conține modulul de date; cîte un segment de date comune din fiecare modul de date comune. Segmentul de program și segmentul de date primesc numele specificat în cartela SEG, iar segmentele de date comune, numele modulelor de date comune pe care le conțin.



Modul de înlănțuire a segmentelor programului trebuie precizat de utilizator, prin cartela TREE, în care vor fi citate numele segmentelor de program.

*Exemplu.* Să se scrie un program COBOL care să realizeze:

— citirea de 5 numere întregi (pozitive și negative), înregistrate pe o cartelă;

— determinarea maximumului acestor numere;

— tipărirea la imprimantă a celor 5 numere și a maximumului.

Fiecare din cele trei operații va fi realizată în cadrul unei unități de program distincte: citirea numerelor în programul principal — numit CITIRE, determinarea maximumului într-un subprogram — numit SMAXIM, iar tipărirea, într-un subprogram — numit AFIȘARE.

În faza de editare a legăturilor programul va fi structurat astfel încât segmentele ce corespund subprogramelor SMAXIM și AFIȘARE să fie încărțate succesiv în aceeași zonă de memorie. Rezolvarea problemei este prezentată în programul XIX.2

```
      JOB SEGMENT2,AN:1100,PN:ANS
      COMPILE COBOL
      ID DIVISION.
      PROGRAM-ID: PIX5.
      ENVIRONMENT DIVISION.
      CONFIGURATION SECTION.
      DATA DIVISION.
      WORKING-STORAGE SECTION.
      01 CARTELA.
         05 NR PIC S9(6) OCCURS 5.
      01 MAXIMUM PIC S9(6).
      01 MAXIMUM-EDITARE PIC +(6)9
      01 I PIC 99
      PROCEDURE DIVISION.
      CITIRE SECTION.
      ACCEPT-PERFORM-STOP.
         ACCEPT CARTELA
         PERFORM SMAXIM
         PERFORM AFIȘARE
         STOP RUN.
      SMAXIM SECTION 60.
      START-MAXIM.
         MOVE NR (1) TO MAXIMUM
         PERFORM CICLU-MAXIM VARYING I FROM 1 BY 1 UNTIL I > 5
         GO TO SMAXIM-EXIT.
      CICLU-MAXIM.
         IF MAXIMUM < NR (I)
            THEN
               MOVE NR (I) TO MAXIMUM.
      SMAXIM-EXIT.
      EXIT.
      AFIȘARE SECTION 70.
      START-AFIȘARE.
         DISPLAY '      SIRUL DE NUMERE:'
         PERFORM SCRIE-NUMERE VARYING I FROM 1 BY 1 UNTIL I > 5
         MOVE MAXIMUM TO MAXIMUM-EDITARE
         DISPLAY '      NUMARUL MAXIM = ' MAXIMUM-EDITARE
         GO TO AFIȘARE-EXIT.
      SCRIE-NUMERE.
         MOVE NR (I) TO MAXIMUM-EDITARE
         DISPLAY '      N( ' I ') = ' MAXIMUM-EDITARE.
      AFIȘARE-EXIT.
      EXIT.
      LINK
      RUN
      11111122222K33333LA4444455555N
      EOJ
      PROGRAMUL XIX.3
```



b. A doua metodă, care permite obținerea unei structuri arborescente cu două nivele, necesită gruparea instrucțiunilor în secțiuni distincte (care să realizeze funcții distincte ale programului). Între secțiuni trebuie să existe o ierarhie, care este pusă în evidență prin intermediul unor *numere de prioritate*, ce sînt asociate secțiunilor. Numerele de prioritate pot lua valori în intervalul [0,99] și sînt indicate în titlul secțiunilor, conform formatului:

*nume-secțiune* SECTION *număr-prioritate*

Pot exista mai multe secțiuni cu același număr de prioritate, iar dacă unor secțiuni nu le-au fost atribuite astfel de numere, compilatorul le atribuie prioritatea 0.

Pe baza numerelor de prioritate, compilatorul va iniția segmentarea programului. Prin compilarea unui program astfel secționat, sînt generate: unul sau mai multe module de program (care este funcție de numerele de prioritate asociate secțiunilor), un modul de date și un număr de module de date comune (care este funcție de numărul fișierelor definite în program). Conținutul și numele atribuite modulelor-de date și de date comune sînt stabilite după aceleași reguli ca și în cazul modulelor de același tip, obținute prin compilarea unui program neseționat.

Un modul de program poate conține: instrucțiunile în limbaj-mașină ce corespund instrucțiunilor din secțiunile cu numere de prioritate mai mici decît 50 — caz în care numele modulului este stabilit adăugînd 01 la primele șase caractere ale numelui programului, sau instrucțiunile în limbaj-mașină ce corespund instrucțiunilor din una sau mai multe secțiuni ce au același număr de prioritate, mai mare decît 49—caz în care numele modulului este stabilit prin adăugarea la primele cinci caractere ale numelui programului a caracterului % și a unui număr format din două cifre, care reprezintă diferența dintre numărul de prioritate al secțiunii (secțiunilor) și 48.

Totodată, compilatorul asociază fiecărui modul de program o cartelă SEG, care indică numele ce va fi atribuit segmentelor de program și de date generate.

Editorul de legături generează din fiecare modul un segment ce corespunde tipului modulului.

Segmentul de program, care conține modulul de program ce corespunde secțiunilor cu numere de prioritate mai mici decît 50, reprezintă *segmentul rădăcină* și, împreună cu segmentul de date, vor primi ca nume, numele modulului de date, iar segmentele de date comune primesc numele modulelor de date comune pe care le conțin.

Numele segmentelor de program care conțin modulele de program ce corespund secțiunilor cu numere de prioritate mai mare decît 49 sînt stabilite astfel: la primele cinci caractere ale numelui programului se adaugă caracterul % și un număr format din două cifre, care reprezintă diferența dintre numărul de prioritate al secțiunilor și 49.

*Exemplu.* Se consideră următorul program COBOL.

ID DIVISION.  
PROGRAM-ID.  
ARBORE.

FILE-CONTROL.  
SELECTFIS-BAND  
ASSIGN TO AAD.



SELECT FIS-DISC  
ASSIGN TO BMT.

PROCEDURE DIVISION.  
PRIMA SECTION.  
DESCHIDE.

OPEN INPUT FIS-BAND  
OUTPUT FIS-DISC

DOI SECTION 7.

TREI SECTION 50.

PATRU SECTION 70.

CINCI SECTION 70.

SASE SECTION 80.

În urma compilării și editării legăturilor acestui program sursă, se obține un program IMT alcătuit din 7 segmente (fig. XIX.1).

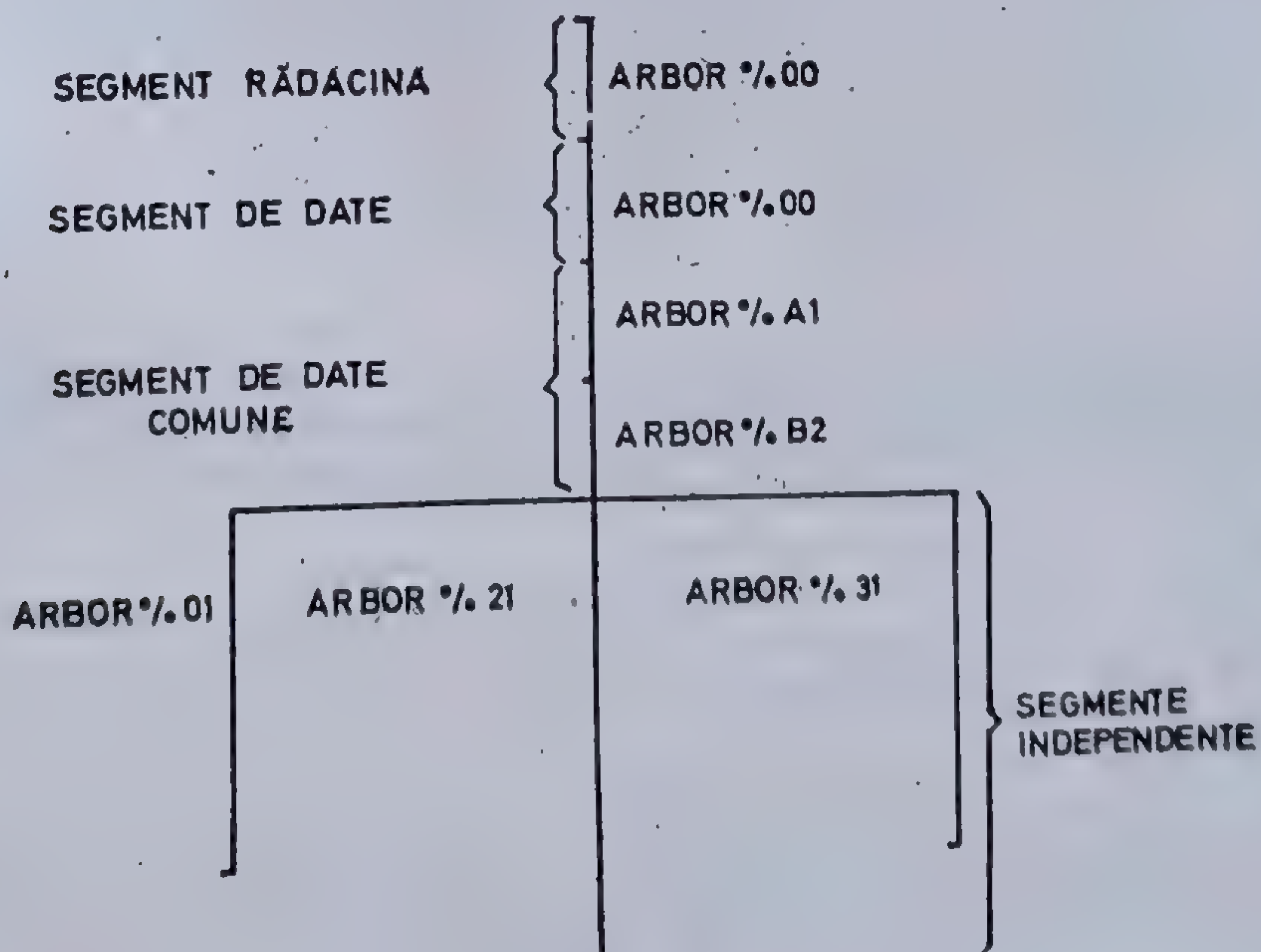


Fig. XIX.1



Segmentele generate sînt dispuse pe două nivele. La primul nivel sînt situate segmentul de program ce corespunde secțiunilor cu numere de prioritate mai mici decît 50, segmentul de date și segmentele de date comune; aceste segmente alcătuiesc *rădăcina* programului. La al doilea nivel sînt situate celelalte segmente de program, numite *segmente independente*, care vor fi încărcate și executate în aceeași zonă de memorie.

Segmentele independente au acces la segmentul de date și la segmentele de date comune. De asemenea, *segmentele independente pot apela execuția unor secvențe de instrucțiuni din segmentul rădăcină.*

La nivelul programului sursă, comunicarea între secțiuni se realizează prin instrucțiuni GO TO sau PERFORM (în care pot fi citate nume de paragrafe sau de secțiuni).

Deoarece segmentele independente nu pot coexista în memorie, utilizarea instrucțiunii PERFORM impune respectarea următoarelor reguli:

- o instrucțiune PERFORM, aflată într-o secțiune cu număr de prioritate  $p$  ( $p > 49$ ) nu poate apela paragrafe din secțiuni cu numere de prioritate mai mari decît  $p$  (în schimb, poate apela paragrafe din secțiuni cu numere de prioritate egale cu  $p$  sau mai mici decît 50);

- o instrucțiune PERFORM aflată într-o secțiune cu număr de prioritate mai mic de 50, poate apela paragrafe conținute într-o singură secțiune, cu număr de prioritate mai mare decît 49.

Pentru optimizarea programului, este indicat ca secțiunile cu numere de prioritate mai mari decît 49 să grupeze instrucțiunile executate mai puțin frecvent; în acest fel se micșorează timpul necesar încărcării în memorie a segmentelor independente ce corespund acestor secțiuni.

*Exemplu.* În programul XIX.3 este prezentată rezolvarea, prin metoda de segmentare prezentată anterior, a problemei enunțate pentru programul XIX.2. Corespunzător celor trei unități de program (CITIRE, SMAXIM, și AFISARE) prezentate în programul XIX.2, diviziunea de prelucrare a programului XIX.3 a fost structurată în trei secțiuni care au același nume ca al unităților de program și numere de prioritate 0, 60 și, respectiv, 70.

### 3. SORTAREA FIȘIERELOR ÎN COBOL

*Sortarea* reprezintă operația de aranjare a articolelor unui fișier într-o anumită ordine (crescătoare sau descrescătoare), în funcție de valorile uneia sau ale mai multor date ale articolului, numite *chei de sortare*. Cheile de sortare sînt stabilite de programator în funcție de prelucrările la care va fi supus fișierul obținut prin sortare.

Elaborarea algoritmului de sortare și transcrierea lui într-un limbaj de programare nu ridică probleme deosebite dacă volumul fișierului de sortat și numărul cheilor de sortare sînt mici, întrucît, în astfel de situații, sortarea se poate realiza prin utilizarea unei zone de lucru ce aparține memoriei centrale.

Creșterea volumului fișierului de sortat și a numărului cheilor de sortare complică algoritmul; deoarece capacitatea limitată a memoriei centrale impune ca zona de lucru să aparțină unei memorii auxiliare (bandă sau disc magnetic).

Pentru simplificarea programării operațiilor de sortare — care apar destul de frecvent în aplicațiile de prelucrare automată a datelor — a fost conceput un ansamblu de module (de sortare), care au fost incluse în Biblio-



teca de Subprograme Standard (BSS) a sistemului de operare. În funcție de anumite informații furnizate de programator, aceste module pot fi apelate de către programul utilitar generator de programe de sortare sau de către programele traducătoare și grupate în segmente de către editorul de legături. După cum apelarea este realizată de generator sau de un program traducător, editorul de legături generează un program IMT sau o secvență de program IMT care, prin execuție, realizează operația de sortare solicitată de programator.

a. Algoritmul de sortare. Operația de sortare, realizată prin intermediul modulelor de sortare, implică prezența a trei fișiere:

- fișierul de intrare, care conține articolele ce trebuie ordonate după anumite criterii (chei);

- fișierul de ieșire, care conține, în urma sortării, articolele fișierului de intrare ordonate conform criteriilor indicate de programator;

- fișierul de manevră, care este utilizat ca zonă de lucru în timpul sortării.

Aceste fișiere prezintă următoarele *caracteristici*:

- suportul poate fi banda sau discul magnetic;

- organizarea trebuie să fie secvențială;

- articolele pot fi de format fix sau variabil, grupate sau nu în blocuri.

Sortarea articolelor unui fișier se realizează în trei *etape*:

- Etapa de constituire a unor șiruri monotone (crescătoare sau descrescătoare) numite *monotonii* și înregistrarea lor în fișierul de manevră.

- Etapa de interclasare a monotonilor în care se obține, în final, o singură monotonie care conține articolele fișierului de intrare ordonate după criteriile stabilite.

Interclasarea monotonilor poate fi realizată prin două *metode*: metoda polifazelor și metoda oscilantă.

În cazul *metodei polifazelor*, fișierul de manevră — care poate avea ca suport banda sau discul magnetic — este structurat în  $N+1$  zone de manevră de aceeași dimensiune:  $N$  este un număr stabilit de programator ( $2 \leq N \leq 6$ ) și se numește *numărul căilor de sortare*.

După cum suportul fișierului de manevră este discul sau banda magnetică, o zonă de manevră reprezintă un număr de cilindri succesivi din cadrul unui volum (pachet de discuri) sau un volum întreg (rolă de bandă).

Dimensiunea unei zone de manevră trebuie să fie cel puțin egală cu dimensiunea fișierului de intrare.

#### Observație

Dacă se alege discul ca suport pentru fișierul de manevră, organizarea acestuia poate fi și secvențial-înlănțuită (în zonă partajată). În acest caz, dimensiunea fișierului de manevră trebuie să fie de 1,5 ori mai mare decât dimensiunea fișierului de intrare.

În cazul *metodei oscilante* fișierul de manevră — care are ca suport discul — este structurat în  $p$  zone de manevră de dimensiuni variabile  $Z_1, Z_2, \dots, Z_p$  ce pot conține  $N, N^2, N^3, \dots, N^p$  *monotonii de dimensiuni fixe*.  $N$  reprezintă numărul căilor de sortare, iar  $p$  este cel mai mic întreg pozitiv care satisface relația:  $M \leq N^p$  ( $M$  reprezintă numărul monotonilor create în prima etapă a sortării).

Fișierul de manevră poate ocupa unul sau mai multe volume, iar dimensiunea sa trebuie să fie de 1,5 ori mai mare decât dimensiunea fișierului de intrare.



— Etapa exploatării fişierului de manevră în care monotonia finală — obţinută în etapa anterioară într-una din zonele fişierului de manevră — este înregistrată pe suportul fişierului de ieşire.

b. **Cheile de sortare.** Ordonarea articolelor fişierului de intrare se realizează în funcţie de conţinutul uneia sau al mai multor *chei de sortare*.

Numărul maxim de chei este 12, iar suma lungimilor zonelor ce le sînt asociate nu trebuie să depăşească 256 locaţii.

Fiecare cheie trebuie să ocupe aceeaşi poziţie în cadrul tuturor articolelor şi să aibă o lungime fixă.

Pentru fiecare cheie se poate stabili sensul sortării: crescător sau descrescător.

Valorile datelor ce reprezintă cheile de sortare pot fi reprezentate în cod EBCDIC, în zecimal despachetat (numere cu semn), în zecimal împachetat, în cod complementar sau în virgulă mobilă simplă sau dublă precizie.

c. **Descrierea operaţiei de sortare.** *Operaţia de sortare* este definită în COBOL prin intermediul unor clauze şi instrucţiuni specifice.

Avantajul utilizării limbajului COBOL pentru realizarea operaţiei de sortare constă în faptul că permite introducerea unor prelucrări speciale în prima şi ultima etapă a acestei operaţii. Astfel, pentru constituirea monotoniei pot fi utilizate articole ce provin dintr-un fişier unic (avînd ca suport bandă sau discul şi organizarea secvenţială), definit în programul COBOL fişier de intrare al sortării, sau articole ce provin din mai multe fişiere. De asemenea, prin exploatarea fişierului de manevră, articolele acestuia pot fi înregistrate într-un fişier unic (avînd ca suport bandă sau discul şi organizarea secvenţială), definit în programul COBOL fişier de ieşire al sortării, sau în mai multe fişiere.

Dacă articolele supuse operaţiei de sortare nu provin din fişierul de intrare al sortării, sau dacă aceste articole nu sînt înregistrate după sortare în fişierul de ieşire al sortării, în programul COBOL este necesar să se definească, în cadrul unor secţiuni distincte ale diviziunii de prelucrare, proceduri de constituire sau de exploatare a fişierului de manevră.

Descrierea în limbajul COBOL a fişierului de manevră se realizează într-un mod particular; celelalte fişiere care intervin în prelucrare se descriu în mod obişnuit.

● În secţiunea INPUT-OUTPUT, fişierului de manevră trebuie să-i corespundă o frază SELECT, al cărui format general este următorul:

$$\text{SELECT } \textit{fişier-manevră} \text{ ASSIGN TO } \textit{idex} \left\{ \begin{array}{l} \text{MT} \\ \text{RD} \\ \text{MD} \\ \text{ZP} \\ \text{AD} \end{array} \right\}.$$

Prin fraza SELECT fişierului de manevră *i* se asociază un nume (*fişier-manevră*), un identificator de exploatare (*idex*), *i* se precizează suportul şi organizarea; opţiunea ZP indică faptul că suportul fişierului de manevră este discul magnetic, iar organizarea este secvenţial-înlănţuită (în zonă partajată).

Dacă suportul indicat pentru fişierul de manevră este banda magnetică, compilatorul COBOL va apela din BSS modulele de sortare care realizează interclasarea monotoniei prin metoda polifazelor; în caz contrar, va apela modulele de sortare care realizează interclasarea monotoniei prin metoda oscilantă.



În secțiunea FILE, descrierea fișierului de manevră este continuată printr-o rubrică SD, al cărui format general este următorul:

SD *fișier-manevră* DATA {RECORD IS  
RECORDS ARE} *nume-articol ...*

Rubrica SD, în care se indică numele articolului (articolelor) fișierului de manevră, este urmată de descrierea articolului (articolelor), realizată prin rubrici obișnuite, care trebuie să pună în evidență cheile de sortare și, eventual, datele asupra cărora se operează prin prelucrările speciale introduse în prima și/sau în ultima etapă a sortării.

#### Observație

Fișierul de manevră nu trebuie să fie citat în instrucțiunile OPEN și CLOSE, deoarece deschiderea și închiderea sa este realizată de modulele de sortare.

● Instrucțiunea SORT realizează apelul modulelor de sortare, cărora le transmite informații referitoare la fișierele care intervin în prelucrare, cheile de sortare și procedurile (prelucrările) ce vor fi executate în prima și/sau în ultima etapă a sortării.

Formatul general al instrucțiunii SORT este următorul:

SORT *fișier-manevră*

{ON {ASCENDING  
DESCENDING} KEY *nume-dată-1, nume-dată-2 ...*}...

{USING *fișier-intrare*  
INPUT PROCEDURE IS *nume-sectiune-1* [THRU *nume-sectiune-2*]}

{GIVING *fișier-ieșire*  
OUTPUT PROCEDURE IS *nume-sectiune-3* [THRU *nume-sectiune-4*]}

Semnificația opțiunilor care apar în formatul general este prezentată în continuare.

● Cheile de sortare sînt indicate prin *nume-dată-1, nume-dată-2 ...* care desemnează date ale articolului fișierului de manevră.

Prioritatea cheilor este definită de ordinea în care acestea sînt citate în instrucțiunea SORT; prima cheie are prioritatea cea mai mare, iar ultima, prioritatea cea mai mică.

Sensul sortării este definit, pentru fiecare cheie, prin opțiunile ASCENDING (crescător) sau DESCENDING (descrescător).

● Opțiunea USING precizează fișierul de intrare al sortării, iar opțiunea GIVING, fișierul de ieșire al sortării. Aceste fișiere trebuie să fie organizate secvențial, iar suportul poate fi discul sau banda magnetică.

Prin execuția unei instrucțiuni SORT, în care figurează opțiunile USING și GIVING, se realizează următoarele operații:

— deschiderea fișierelor citate în opțiunile USING și GIVING, precum și a fișierului de manevră;

— transferul succesiv al articolelor fișierului citat în opțiunea USING în zona de lucru din memoria centrală, constituirea monotonilor, și înregistrarea acestora pe suportul fișierului de manevră;



- interclasarea monotoniilor create și obținerea monotoniei finale;
- transferul succesiv al articolelor monotoniei finale în fișierul citat în opțiunea GIVING;
- închiderea fișierelor citate în opțiunile USING și GIVING, precum și a fișierului de manevră.

*Exemplu.* În programul XIX.4 se prezintă descrierea în limbajul COBOL a operației de sortare a articolelor fișierului DESFACERI, creat prin execuția programului XVIII.1. Sortarea este realizată crescător, după valorile datei COD-MAGAZIN, prin metoda oscilantă, fișierul de ieșire fiind înregistrat pe disc.

● Procedura de intrare (INPUT PROCEDURE) grupează, în cadrul uneia sau mai multor secțiuni consecutive (deșemnate în formatul instrucțiunii SORT, prima, prin *nume-secțiune-1*, iar ultima, prin *nume-secțiune-2*) instrucțiunile prin care se realizează constituirea fișierului de manevră.

```

      JOB SORTARE1,AN:II00,PN:ANS
      COMPILE COBOL
      ID DIVISION.
      PROGRAM-ID. PIX6.
      ENVIRONMENT DIVISION.
      CONFIGURATION SECTION.
      INPUT-OUTPUT SECTION.
      FILE-CONTROL.
          SELECT FISIER-INTRARE ASSIGN TO AMT.
          SELECT FISIER-MANEVRA ASSIGN TO BMT
          SELECT FISIER-IESIRE ASSIGN TO CAD.
      DATA DIVISION.
      FILE SECTION.
      FD FISIER-INTRARE RECORDING F
          LABEL RECORD STANDARD BLOCK 15 RECORDS.
      01 ART-I PIC X(61).
      SD FISIER-MANEVRA DATA RECORD IS ART-M.
      01 ART-M.
          05 COD-MAGAZIN PIC 99.
          05 FILLER PIC X(59).
      FD FISIER-IESIRE RECORDING F
          LABEL RECORD STANDARD BLOCK 15 RECORDS.
      01 ART-E PIC X(61).
      PROCEDURE DIVISION.
      START-SORT-STOP.
          SORT FISIER-MANEVRA
              ON ASCENDING KEY COD-MAGAZIN
              USING FISIER-INTRARE
              GIVING FISIER-IESIRE
          STOP RUN.
      LINK.
      ASSIGN A,DVT:MT,VS:MT1MAN
      LABEL A,FN:'DESFACERI'
      INIT DVT:MT,VS:MT2MAN
      INIT DVT:MT,VS:MT3MAN
      ASSIGN B,DVT:MT,VS:MT1MAN+MT2MAN+MT3MAN
      LABEL B,FN:'MANEVRA'
      LABEL C,FN:'DESFACERI-SORT'
      RUN
      EOJ
  
```

PROGRAMUL XIX.4.



Articolele ce vor alcătui fișierul de manevră pot proveni din unul sau din mai multe fișiere. În situațiile în care fișierul de manevră nu trebuie să conțină toate articolele acestor fișiere sau când structura articolelor acestor fișiere diferă de cea a articolelor fișierului de manevră, procedura de intrare trebuie să realizeze selectare și/sau constituirea lor conform structurii articolelor fișierului de manevră. Articolele astfel stabilite sînt furnizate succesiv modulelor de sortare (în vederea grupării lor în monotonii) prin intermediul instrucțiunii RELEASE, al cărui *format general* este:

**RELEASE** *nume-articol* [**FROM** *nume-dată*]

unde:

- *nume-articol* reprezintă numele unui tip de articol al fișierului de manevră;
- opțiunea FROM este similară cu opțiunea FROM a instrucțiunii WRITE.

● Procedura de ieșire (OUTPUT PROCEDURE) conține instrucțiunile prin care se prelucurează articolele fișierului de manevră. Aceste instrucțiuni sînt grupate în cadrul uneia sau mai multor secțiuni consecutive (prima fiind desemnată prin *nume-secțiune-3*, iar ultima, prin *nume-secțiune-4*).

Accesul la articolele fișierului de manevră se realizează prin intermediul instrucțiunii RETURN, al cărui *format general* este:

**RETURN** *fișier-manevră* [**INTO** *nume-dată*] **AT END** *instrucțiune*.

Execuția unei instrucțiuni RETURN realizează aceleași operații ca și instrucțiunea READ pentru acces secvențial.

Prelucrarea articolelor fișierului de manevră constă, în general, în transferarea lor — totală sau parțială — cu sau fără modificarea structurii, în unul sau în mai multe fișiere.

Prin execuția unei instrucțiuni SORT în care figurează opțiunile INPUT PROCEDURE și OUTPUT PROCEDURE se realizează următoarele operații:

- deschiderea fișierului de manevră;
- execuția instrucțiunilor procedurii de intrare;
- interclasarea monotoniiilor;
- execuția instrucțiunilor procedurii de ieșire;
- închiderea fișierului de manevră.

După efectuarea acestor operații, execuția programului continuă cu instrucțiunea ce urmează instrucțiunii SORT.

#### Observații

- În instrucțiunea SORT pot fi utilizate combinate opțiunile USING sau GIVING cu opțiunile INPUT PROCEDURE sau OUTPUT PROCEDURE.
- Procedurile de intrare și de ieșire nu trebuie să conțină secțiuni comune.
- Procedurile de intrare și de ieșire nu trebuie să conțină instrucțiuni de salt (GO TO, PERFORM) în afara lor.
- Secțiunile diviziunii de prelucrare care nu aparțin procedurilor de intrare sau de ieșire nu trebuie să conțină instrucțiuni de salt (GO TO, PERFORM) în procedurile de intrare sau de ieșire.
- Procedura de intrare trebuie să conțină cel puțin o instrucțiune RELEASE, iar procedura de ieșire, cel puțin o instrucțiune RETURN.
- În procedurile de intrare și de ieșire nu trebuie să apară instrucțiunea SORT.



*Exemplu.* Programul XIX.5 creează fișierul secvențial-indexat BIBLIOTECA, din articolele fișierului pe cartele FIȘIER-CARTI (vezi programul XVIII.5). Comparativ cu programul XVIII.5. acest program realizează în plus, ordonarea articolelor fișierului pe cartele, crescător după valorile datei COD-CARTE. Citirea articolelor fișierului FIȘIER-CARTI este realizată în procedura INTRARE, iar scrierea articolelor în fișierul BIBLIOTECA, în procedura IEȘIRE.

#### 4. ÎNTOCMIREA RAPOARTELOR PRIN INTERMEDIUL EDITORULUI COBOL

a. **Generalități.** Un raport reprezintă un ansamblu de linii tipărit la imprimantă sau înregistrat pe suport magnetic, în vederea unei tipăriri ulterioare.

În general, realizarea unui raport implică prezența unui *fișier de intrare* ale cărui articole sînt ordonate, crescător sau descrescător, după valorile uneia sau mai multor date, numite *variabile de control*. Articolele consecutive ale fișierului de intrare pentru care valorile unei variabile de control sînt egale alcătuiesc o *grupă de control*. În cadrul raportului fiecărui articol și/sau fiecărei grupe de control din fișierul de intrare îi poate fi asociat un număr de linii care alcătuiesc o grupă de editare. Grupele de editare pot fi de mai multe tipuri și au, în general, semnificația următoare:

— *REPORT HEADING (RH)* — conține elementele de identificare a raportului (titlul raportului), fiind tipărită o singură dată, la începutul raportului;

— *PAGE HEADING (PH)* — conține elementele de identificare a unei pagini, fiind tipărită la începutul fiecărei pagini a raportului;

— *DETAIL (DE)* — conține valorile datelor ce aparțin unui articol din fișierul de intrare sau valori rezultate prin prelucrarea acestor date, fiind tipărită după citirea și prelucrarea fiecărui articol din fișierul de intrare;

— *CONTROL HEADING (CH)* — conține elementele de identificare a unei grupe de control din fișierul de intrare, fiind tipărită înaintea primei grupe de editare de tip *DETAIL* și atunci cînd valoarea unei variabile de control se schimbă;

— *CONTROL FOOTING (CF)* — conține totaluri ale valorilor datelor ce aparțin unei grupe de control din fișierul de intrare, fiind tipărită atunci cînd valoarea unei variabile de control se schimbă,

— *PAGE FOOTING (PF)* — conține totalurile ce corespund unei pagini, fiind tipărită la sfîrșitul fiecărei pagini a raportului;

— *REPORT FOOTING (RF)* — conține totaluri generale, fiind tipărită o singură dată, la sfîrșitul raportului.

##### Observație

În cazul în care utilizatorul indică o paginare a raportului (definind, deci, în program grupe de editare de tip PH și, eventual, PF), pentru dispunerea în cadrul paginilor a grupelor de editare și a liniilor ce le corespund, editorul COBOL rezervă raportului două registre speciale, identificate prin cuvinte cheie: *PAGE-COUNTER*, utilizat pentru numerotarea paginilor raportului, și *LINE-COUNTER* utilizat pentru numerotarea liniilor în cadrul paginilor. Aceste registre, ale căror valori sînt, în general, gestionate de editorul COBOL, pot fi referite în program.



Pentru exemplificarea tipurilor de grupe de editare ale unui raport, se consideră exemplul prezentat în continuare.

Fie un fișier pe cartele ale cărui articole conțin date referitoare la valorile producțiilor obținute de secțiile unor întreprinderi: cod minister, cod centrală, cod întreprindere, cod secție, valoare. Articolele sînt ordonate crescător după valorile datelor cod minister, cod centrală și cod întreprindere.

În tabelul XIX.1 este prezentat un raport obținut prin prelucrarea acestui fișier. Raportul conține toate tipurile de grupe de editare și ocupă mai multe pagini. Ca variabile de control au fost stabilite datele cod minister, cod centrală și cod întreprindere. Titlul raportului este tipărit pe prima pagină a raportului.

TABELUL XIX.1

SITUAȚIE } RH				PAG 2 } PH
MINISTER	CENTRALA	INTREPR.	SECȚIE	TOTAL
01	11	222	33	00100 } DE
			44	00200
		TOTAL	INTREPR	300 } CF — CODINT
01	11	333	55	00300
		TOTAL	INTREPR	300
	TOTAL	CENTRALA		600 } CF — CODCEN
01	22	333	66	00100
		TOTAL	INTREPR	100
01	22	444	11	00200
			22	00300
			33	00400
			44	00500
		TOTAL	INTREPR	1400
	TOTAL	CENTRALA		1500
TOTAL	TOTAL			00002100 } CF — CODMIN
02	11	222	33	00110
		TOTAL	INTREPR	110

				PAG 3
MINISTER	CENTRALA	INTREPR	SECȚIE	TOTAL
02	11	33	44	00120
			55	00130
			66	00140
		TOTAL	INTREPR	390
	TOTAL	CENTRALA		500
02	22	444	11	00150
		TOTAL	INTREPR	150
	TOTAL	CENTRALA		150
TOTAL	TOTAL			00000650

TOTAL GENERAL 2750 } RF



b. **Descrierea rapoartelor.** Liniiile ce alcătuiesc un raport reprezintă articole ale unui fișier definit în programul COBOL ca fișier de ieșire.

În general, fișierul de ieșire are ca suport hîrtia de imprimat; este însă posibilă și utilizarea altui suport. Acestui fișier trebuie să-i fie asociată o descriere obișnuită (o rubrică SELECT) în paragraful FILE-CONTROL. Rubrica FD, prin care se continuă descrierea fișierului de ieșire în secțiunea FILE, trebuie să indice numele raportului, prin clauza REPORT:

$$\left\{ \begin{array}{l} \text{REPORT IS} \\ \text{REPORT ARE} \end{array} \right\} \text{nume-raport-1 } [, \text{nume-raport-2}] \dots$$

și eventual lungimea maximă a articolelor (liniilor), prin clauza RECORD, ținînd cont și de caracterul de salt. Rubrica FD nu trebuie să fie urmată de descrierea articolelor fișierului de ieșire (acestea, reprezentînd linii ale raportului, vor fi descrise în secțiunea REPORT).

Descrierea structurii fizice a raportului și a organizării acestuia este realizată în secțiunea REPORT, prin intermediul rubricii RD (REPORT DESCRIPTION), al cărei format este:

RD *nume-raport*

$$\left\{ \left\{ \begin{array}{l} \text{CONTROL IS} \\ \text{CONTROLS ARE} \end{array} \right\} \left\{ \begin{array}{l} \text{FINAL } \textit{nume-data-1} [, \textit{nume-data-2}] \dots \\ \text{FINAL } \textit{nume-data-1} [, \textit{nume-data-2}] \dots \end{array} \right\} \right\}$$

$$\left[ \text{PAGE} \left\{ \begin{array}{l} \text{LIMIT IS} \\ \text{LIMITS ARE} \end{array} \right\} \text{nr } 1 \left\{ \begin{array}{l} \text{LINE} \\ \text{LINES} \end{array} \right\} \right]$$

[HEADING *nr 2*]

[FIRST DETAIL *nr 3*]

[LAST DETAIL *nr 4*]

[FOOTING *nr 5*]

Numele raportului trebuie să fie unic (deoarece nu poate fi calificat) și trebuie să coincidă cu cel indicat în rubrica FD prin care este descris fișierul de ieșire.

Clauzele rubricii RD sînt opționale și pot fi scrise în orice ordine. Semnificația lor este prezentată în continuare.

**Clauza CONTROL** indică variabilele de control stabilite pentru realizarea raportului, precum și prioritatea, în ordine descrescătoare, a efectuării controalelor. Controalele constau în testarea, înaintea generării unei grupe de editare de tip DE, a valorilor variabilelor de control. Testarea se efectuează începînd cu variabila cu prioritatea cea mai mare (*nume-data-1*). Dacă testarea indică schimbarea valorii uneia dintre variabilele de control, tipărirea grupei de editare de tip DE ce corespunde ultimului articol citit din fișierul de intrare este suspendată temporar pentru:

— tipărirea grupei de editare de tip CF ce corespunde grupei de control din fișierul de intrare determinată de valoarea variabilei de control testată, precum și a grupelor de editare de tip CF ce corespund grupelor de control determinate de valorile variabilelor de control a căror prioritate este inferioară variabilei de control testată;

— tipărirea grupelor de editare de tip CH ce corespund noilor grupe de control determinate de noile valori ale variabilei de control testată și de valorile variabilelor de control cu priorități inferioare acesteia.



Opțiunea FINAL indică nivelul de control maxim, determinând întreruperi temporare pentru tipărirea unor grupe de editare la începutul raportului (înaintea tipăririi primei grupe de editare de tip DE) sau la sfârșitul raportului (după tipărirea ultimei grupe de tip DE).

În figura XIX.2 este prezentată schema logică de principiu care indică modul de efectuare a controalelor, considerând raportul din tabelul XIX.1.

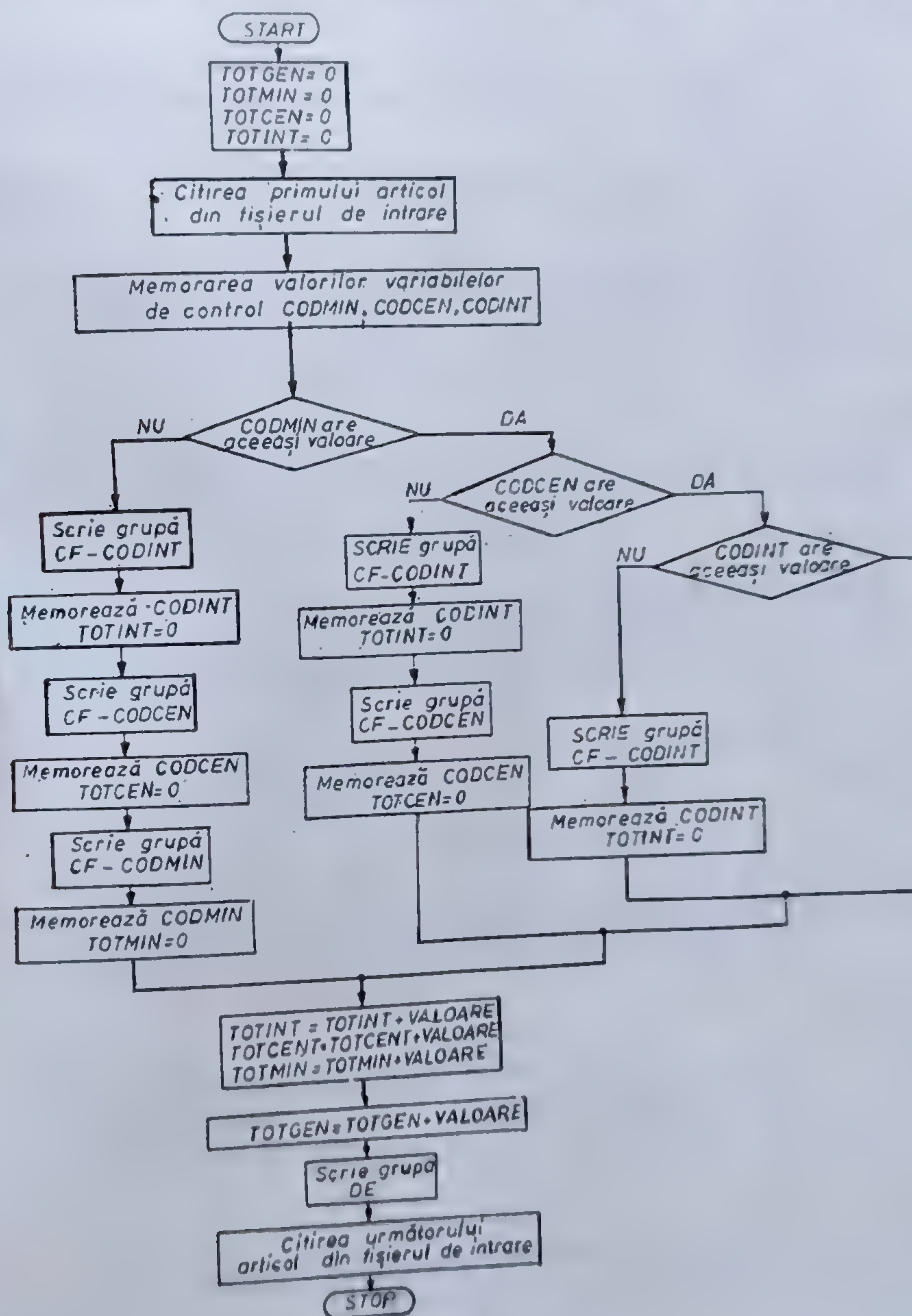


Fig. XIX.2



**Clauza PAGE LIMIT** indică dimensiunea și structura unei pagini a raportului. Semnificația opțiunilor acestei clauze este următoarea: •

— *nr1* indică numărul maxim de linii ce vor fi tipărite pe o pagină a raportului;

— *nr2* indică poziția în cadrul paginii a primei linii ce aparține unei grupe de editare de tip RH sau PH; dacă *nr2* este omis el va fi considerat implicit egal cu *nr1*;

— *nr3* indică poziția în cadrul paginii a primei linii ce aparține unei grupe de editare de tip DE, CH sau CF; dacă *nr3* este omis va fi considerat implicit egal cu *nr2*;

— *nr4* indică poziția în cadrul paginii a ultimei linii ce aparține unei grupe de editare de tip DE sau CF; dacă *nr4* este omis, el va fi considerat implicit egal cu *nr5*;

— *nr5* indică poziția în cadrul paginii a ultimei linii ce aparține unei grupe de editare de tip CF; dacă *nr5* este omis, el va fi considerat implicit egal cu *nr4*.

#### Observație

Valorile lui *nr1*, *nr2*, *nr3*, *nr4*, *nr5* trebuie să satisfacă relația:

$$1 \leq nr2 \leq nr3 \leq nr4 \leq nr5 \leq 999$$

• Pentru un raport care conține toate tipurile de grupe de editare valorile lui *nr1*, *nr2*, *nr3*, *nr4*, *nr5* trebuie stabilite ca în figura XIX.3.

• *nr3* trebuie obligatoriu indicat dacă raportul conține grupe de editare de tip PH.

• *nr4* (sau *nr5*) trebuie indicat dacă raportul conține grupe de editare de tip PF.

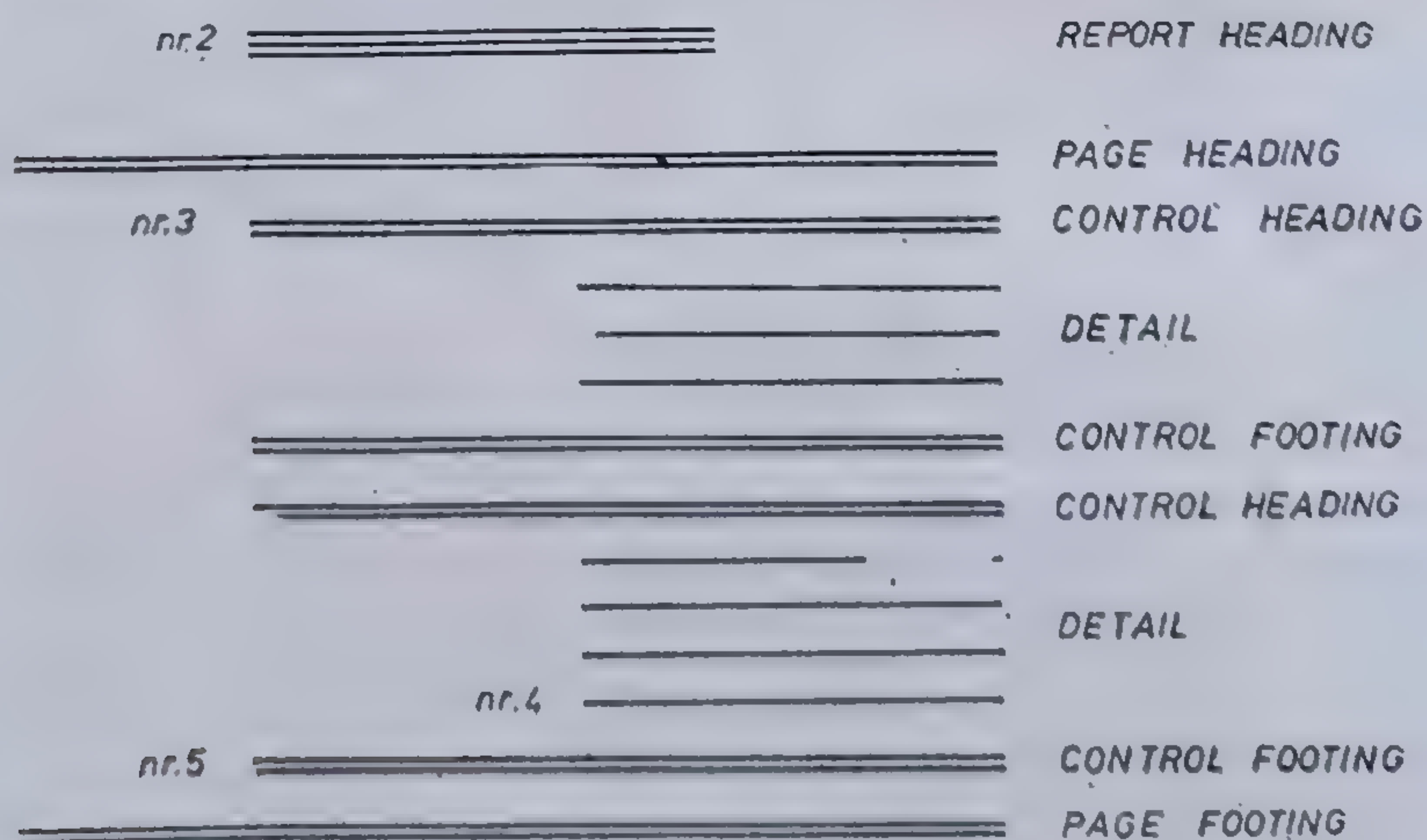


Fig. XIX.3



• Rubrica RD trebuie să fie urmată de rubricile care descriu grupele de editare, liniile ce compun aceste grupe, precum și datele conținute de aceste linii. Există două formate generale pentru aceste rubrici:

— format general 1, utilizat pentru descrierea grupelor de editare:

01 [nume-data-1]

TIPE IS {

{	REPORT HEADING	}	
{	RH	}	
{	PAGE HEADING	}	
{	PH	}	
{	CONTROL HEADING	}	{ nume-data-2 }
{	CH	}	{ FINAL }
{	DETAIL	}	
{	DE	}	
{	CONTROL FOOTING	}	{ uume-data-3 }
{	CF	}	{ FINAL }
{	PAGE FOOTING	}	
{	PF	}	
{	REPORT FOOTING	}	
{	RF	}	

}

[ LINE NUMBER IS { nr1 [ON NEXT PAGE] } ]

[ ON NEXT PAGE ]

[ PLUS nr2 ]

[ NEXT GROUP IS { nr3 } ]

[ PLUS nr4 ]

[ ON NEXT PAGE ]

— format general 2, utilizat pentru descrierea liniilor și a datelor (acest format poate fi însă utilizat și pentru descrierea grupelor de editare care conțin o singură linie; aceasta impune însă utilizarea clauzelor TYPE și NEXT GROUP):

număr nivel [nume-data-1]

[BLANK WHEN ZERO]

[ LINE NUMBER IS { nr1 [ON NEXT PAGE] } ]

[ ON NEXT PAGE ]

[ PLUS nr 2 ]

[ { PICTURE } IS șablon ]

[ PIC ]

[VALUE IS literal]



[COLUMN NUMBER IS *nr3*]

[GROUP INDICATE]

[SUM *nume-data-2* [, *nume-data-3*] ...]

[SOURCE IS *nume-data-4*].

Într-o rubrică de format 1 trebuie indicat obligatoriu numărul de nivel 01 și clauza TYPE; *nume-data 1* trebuie indicat, în general, pentru grupele de tip DE.

Clauza TYPE indică tipul grupei de editare. Opțiunile RH, PH, CH, DE, CF, PF, RF au semnificația precizată anterior. Opțiunile *nume-data-2* și *nume-data-3* indică faptul că grupele de editare de tip CH și CF vor fi tipărite la schimbarea valorilor variabilelor de control *nume-data-2* și *nume-data-3* (acestea trebuie deci precizate și în clauza CONTROL a rubricii RD). Opțiunea FINAL indică faptul că grupele de editare de tip CH și CF vor fi tipărite o singură dată, aceste grupe fiind asociate primei și, respectiv, ultimei grupe de control din fișierul de intrare.

#### Observație

Grupele de editare de tip RH, PH, PF și RF nu pot fi definite decât o singură dată în cadrul unui raport.

Rubricile ce descriu grupele de editare ale unui raport trebuie să apară, obligatoriu, în ordinea:

REPORT HEADING  
PAGE HEADING  
CONTROL HEADING FINAL  
CONTROL HEADING *nume-data-1*

CONTROL HEADING *nume-data-n*  
DETAIL  
CONTROL FOOTING *nume-data-n*

CONTROL FOOTING *nume-data-1*  
CONTROL FOOTING FINAL  
PAGE FOOTING  
REPORT FOOTING

Clauza LINE indică poziția (absolută sau relativă) în care va fi tipărită prima linie a grupei de editare.

Semnificația opțiunilor acestei clauze este:

— *nr1* — linia va fi tipărită pe pagina curentă, în poziția indicată prin *nr1* ( $1 \leq nr1 \leq 999$ );

— *nr1 ON NEXT PAGE* — linia va fi tipărită pe pagina următoare, în poziția indicată prin *nr1*;



— *ON NEXT PAGE* — linia va fi tipărită pe pagina următoare, în poziția indicată prin opțiunea clauzei *PAGE LIMIT* ce corespunde grupei de editare ce conține linia;

— **PLUS nr2** — linia va fi tipărită pe pagina curentă, în poziția relativă (adică, față de linia tipărită anterior) indicată prin **nr2** ( $2 \leq \text{nr2} \leq 999$ )

### Observatii

- nr1 trebuie să aibă valori în intervalul definit de clauza PAGE LIMIT.
- Clauza LINE trebuie să figureze în fiecare rubrică ce descrie o linie a unei grupe de editare.
- Opțiunea ON NEXT PAGE poate fi utilizată numai în rubrica de descriere a unei grupe de editare de tip CH, DE, CF, RF.

Utilizarea opțiunii *PLUS nr2* în rubrica de descriere a unei grupe de editare impune ca această rubrică să fie precedată de o rubrică în care să nu figureze această opțiune.

**Clauza NEXT GROUP** indică numărul de linii „libere” ce vor fi lăsate după scrierea ultimei linii a grupei de editare. Prin prelucrarea acestei clauze, în funcție de opțiunea utilizată, se realizează:

— atribuirea valorii *nr3* registrului LINE COUNTER, dacă este utilizată opțiunea *nr3*;

— valoarea registrului LINE COUNTER este mărită cu *nr4*, dacă este utilizată opțiunea PLUS *nr4*;

— sînt tipărite grupele de editare PF pentru pagina curentă și PH pentru pagina următoare, dacă este utilizată opțiunea ON NEXT PAGE.

## Observatie

Clauza NEXT GROUP nu poate fi specificată în rubricile ce descriu grupe de editare de tip PH, PF și RF

În general, formatul 2 este utilizat pentru descrierea unei linii, dacă este prezentă clauza LINE, sau a unei date ce aparține unei linii, dacă nu este prezentă clauza LINE.

Semnificația și utilizarea clauzelor BLANK, LINE, PICTURE, VALUE sînt cunoscute; pentru celelalte clauze ele sînt prezentate în continuare.

*Clauza COLUMN* indică, prin  $nr3$  ( $1 \leq nr3 \leq 132$ ), numărul coloanei din cadrul liniei începînd de la care vor fi înregistrate (de la stînga la dreapta) valorile datei.

În exemplul următor se prezintă descrierea unei linii.

01 LINIE.

01	LINE.			
02	COLUMN	5	VALUE 'ABCD'	PIC X(5).
02	COLUMN	20	VALUE 12.5	PIC 99V9.
02	COLUMN	12	VALUE ALL '*'	PIC X(4).

02 COLUMN 20 VALUE 12.5  
02 COLUMN 12 VALUE ALL '\*' PIC X(4).

Imaginea zonei asociate acestei linii este:

Imaginea zonei asociate acestor numere este:

				A	B	C	D				*	*	*	*					1	2	5
--	--	--	--	---	---	---	---	--	--	--	---	---	---	---	--	--	--	--	---	---	---

Notă: Se compun unia nu li s-au asociat numere.

			-		A	B	C	D											
--	--	--	---	--	---	---	---	---	--	--	--	--	--	--	--	--	--	--	--

Se observă faptul că datelor ce compun linia nu li s-au asociat nume (considerînd că acestea nu vor fi referite) și că datele nu au fost definite în ordinea în care apar în cadrul liniei. De asemenea, se poate remarca faptul

241



că această clauză nu poate figura decât în rubrica de descriere a unei date elementare.

**Clauza GROUP** indică faptul că datei *i* se vor atribui valori numai la prima tipărire a liniei în care figurează data (care aparține unei grupe de editare de tip DE) sau la prima tipărire a liniei pe o pagină nouă.

**Clauza SUM** indică faptul că data reprezintă un contor (de însumare) pentru obținerea totalurilor grupelor de control și poate figura numai în rubrica de descriere a unei date elementare ce aparține unei grupe de editare de tip CF.

Datele *nume-data-2*, *nume-data-3*, ... trebuie să fie numerice, pot fi descrise în secțiunile FILE, WORKING și LINKAGE și trebuie să fie citate în clauzele SOURCE ale unor rubrici ce descriu date ce aparțin grupelor de editare de tip DE.

**Clauza SOURCE** indică faptul că datei *i* se va atribui valoarea datei *nume-data-4*, care poate fi unul dintre registrele PAGE COUNTER, LINE COUNTER sau o dată descrisă în diviziunea de date. Înregistrarea valorii în zona asociată datei se realizează conform regulilor de transfer ale instrucțiunii MOVE

c. **Instrucțiuni pentru generarea rapoartelor.** Trei instrucțiuni sînt disponibile (și obligatorii) pentru a genera un raport: INITIATE, GENERATE, TERMINATE.

**Instrucțiunea INITIATE.** Inițiază operația de generare a unui sau mai multor rapoarte, adică inițializează cu valoarea zero toate contoarele definite prin clauza SUM, registrele LINE COUNTER și PAGE COUNTER.

Formatul general al acestei instrucțiuni este:

INITIATE *nume-raport-1* [, *nume-raport-2*] ...

**Instrucțiunea GENERATE.** Are următorul format:

GENERATE { *nume-raport*  
          *nume-grupă-DETAIL* }

și indică, în general, realizarea următoarelor operații:

— atribuirea de valori registrelor LINE-COUNTER și PAGE-COUNTER în vederea tipăririi grupelor de editare de tip RH și/sau PH;

— testarea valorilor variabilelor de control pentru generarea grupelor de editare de tip CH și/sau CF;

— obținerea totalurilor în contoarele definite prin clauza SUM și inițializarea acestora la schimbarea grupelor de control;

— generarea și tipărirea numai la prima execuție a grupelor de editare de tip RH, PH și a grupelor de editare de tip CH, în ordinea indicată de ierarhia variabilelor de control:

CONTROL HEADING FINAL  
CONTROL HEADING *nume-data-1*  
.  
.  
.  
CONTROL HEADING *nume-data-*

unde *n* reprezintă numărul variabilelor de control;

— generarea și tipărirea grupe de editare de tip DE, citată în instrucțiune;



— generarea și tipărirea, în momentul schimbării valorii unei variabile de control, a grupelor de editare de tip CF, în ordinea inversă ierarhiei variabilelor de control;

CONTROL FOOTING *nume-data-m*

CONTROL FOOTING *nume-data-n*

(unde *m* reprezintă numărul de ordine al variabilei de control a cărei valoare s-a schimbat), a grupelor de editare de tip CH pentru noile grupe de control:

CONTROL HEADING *nume-data-m*

CONTROL HEADING *nume-data-n*

și a grupei de editare de tip DE; aceasta din urmă va fi tipărită numai dacă în instrucțiunea GENERATE este citat numele său.

#### Observație

Dacă în instrucțiunea GENERATE este citat numele unui raport acesta va fi un raport simplificat, care nu conține grupe de editare de tip DE.

**Instrucțiunea TERMINATE.** Încheie operația de generare a unui sau a mai multor rapoarte.

Format general:

**TERMINATE** *nume-raport-1* [, *nume-raport-2*] ....

Prin execuția sa se realizează (în cazul unui singur raport):

- generarea și tipărirea ultimelor grupe de editare de tip CF ce corespund ultimelor valori ale variabilelor de control;
- generarea și tipărirea grupei de editare de tip RF.

#### Observație

Editorul COBOL realizează însumarea automată a valorilor datelor fișierului de intrare numai în contoarele definite prin clauza SUM. Dacă, de exemplu, raportul trebuie să conțină totaluri și în alte grupe de editare decât cele de tip CF, dacă înainte de tipărirea unei grupe de editare trebuie efectuate anumite prelucrări sau dacă trebuie anulată tipărirea unei grupe de editare aceste operații trebuie descrise în secțiuni declarative. O astfel de secțiune trebuie să fie precedată de o frază USE de forma următoare:

**USE BEFORE REPORTING** *nume-grupă-editare*.

Secțiunea conține instrucțiuni COBOL (exceptând instrucțiunile GENERATE, INITIATE, TERMINATE), grupate în unul sau mai multe paragrafe

Programul XIX.6 ilustrează editarea raportului prezentat în tabelul XIX.1.



## PROBLEME

1. Să se scrie un subprogram COBOL care, transmitându-i-se din programul apelat un număr format din două cifre, să transmită programului apelant scrierea în litere acestui număr (exemplu: 87 — OPTZECI ȘI ȘAPTE).
2. Să se scrie un program COBOL care să realizeze următoarele operații:
  - citirea de pe o cartelă a 10 numere;
  - ordonarea crescătoare a acestor numere;
  - tipărirea șirului obținut în urma ordonării.

Programul va fi scris în două variante, corespunzător metodelor de segmentare oferite de limbajul COBOL, astfel încât ordonarea și tipărirea să se realizeze în subprograme (secțiuni) distincte, care în timpul execuției, să se reacopere

3. Să se scrie un program COBOL pentru sortarea crescătoare a articolelor fișierului BIBLIOTECA, creat în programul XVIII.5, după valorile datei AUTOR.
4. Să se consulte fișierul creat la punctul 3 și să se editeze, prin intermediul editorului COBOL, un raport care să indice titlul cărților și autorul.



## CAPITOLUL XX

### PROGRAMUL DE INVENTARIERE A FIȘIERELOR ȘI VOLUMELOR

În cadrul aplicațiilor de prelucrare automată a datelor apare frecvent necesitatea obținerii unor situații care să furnizeze informații referitoare la conținutul unui fișier (inventarul fișierului) sau al unui volum (rezumatul volumului).

În general, *inventarierea unui fișier* se efectuează după crearea sau actualizarea acestuia, în scopul verificării acestor operații, iar *rezumarea unui volum*, pentru cunoașterea stării de ocupare a acestuia (numărul fișierelor conținute, implantarea lor în cadrul volumului, zonele neocupate etc.).

Aceste operații sînt realizate prin intermediul programului de inventariere, catalogat în BUS sub numele INVFIȘ.

#### 1. INVENTARIEREA FIȘIERELOR

Operația de inventariere constă în tipărirea la imprimantă a tuturor înregistrărilor unui fișier — *inventariere totală* — sau a unui anumit număr de înregistrări — *inventariere parțială*.

Forma de imprimare poate fi:

- *alfanumerică*, cînd fiecărui bait al înregistrării *i* se asociază caracterul imprimabil care îi corespunde;
- *hexazecimală*, cînd fiecărui bait al înregistrării *i* se asociază cifrele hexazecimale care îi corespund;
- *mixtă*, cînd înregistrarea este imprimată atît alfanumeric cît și hexazecimal.

Fișierele care fac obiectul operației de inventariere pot avea modul de organizare:

- *nedefinit* (suport: bandă magnetică, disc magnetic);
- *secvențial* (suport: cartelă, bandă magnetică, disc magnetic);
- *secvențial-indexat* (suport: disc magnetic);
- *selectiv* (suport: disc magnetic).

În funcție de suport și mod de organizare, fișierele pot fi prelucrate la nivel de: zonă, bloc, sector, căsuță sau articol, iar unitatea de imprimare poate fi: blocul, sectorul sau articolul.

În momentul tipăririi, unității de imprimare *i* se asociază un antet care indică poziția acesteia în cadrul fișierului (tabelul XX.1)

Ordinele prin care este definită operația de inventariere sînt: DUMP, DFLE și KEY.

Aceste ordine (ca de altfel toate ordinele adresate programelor utilitare) sînt perforate pe cartele conform următorului format:

% Ordin Arg1, Arg2, ..., Argn



TABELUL XX.1

Mod de organizare	Suport	Antet	Observații
nedefinită	bandă magnetică	ZnnBmmmm	nn — numărul zonei mmmm — numărul blocului în cadrul zonei
	disc magnetic	ZnnSmmmmm	nn — numărul zonei mmmm — numărul sectorului în cadrul zonei
secvențială	carte	Cnnnnn	nnnn — numărul de ordine al cartei în cadrul fișierului
	bandă magnetică	ZnnBmmmmRxxx	nn — numărul zonei mmmm — numărul blocului/sectorului în cadrul zonei xxx — numărul articolului în cadrul blocului/sectorului
secvențială indexată	disc magnetic	Rnnnnn $\left\{ \begin{matrix} P \\ D \end{matrix} \right\}$	nnnn — numărul de ordine al articolului în cadrul fișierului P — articolul se află în partea principală D — articolul se află în partea de depășire
	disc magnetic	CnnRmmmm $\left\{ \begin{matrix} P \\ D \end{matrix} \right\}$ [Mxxxxx]	nn — numărul căsuței mmmm — numărul de ordine al articolului în cadrul căsuței P — articolul se află în partea principală D — articolul se află în partea de depășire xxxxx — valoarea contorului de actualizare



Scrierea ordinelor pe formularul de programare se realizează conform următoarelor reguli:

- în coloana 1 se înregistrează caracterul %;
- din coloana 3 se înregistrează ordinul;
- între ordin și argumente trebuie să existe un spațiu;
- argumentele sînt separate prin virgulă și nu pot depăși coloana 71;
- în cazul în care argumentele nu încap pe o singură linie, continuarea pe linia următoare se realizează astfel:
  - în linia în care este specificat ordinul se scrie pînă la coloana 71 inclusiv, iar în coloana 72 se înregistrează caracterul \*;
  - în linia următoare se înregistrează caracterul % în prima coloană, iar scrierea argumentelor continuă în coloana 11;
- în cazul continuării, argumentele nu pot fi despărțite.

*Ordinul DFLE* indică parametrii de identificare a fişierului şi, în cazul inventarierii parţiale, condiţiile de început ale inventarierii.

*Format general:*

$$\left\{ \begin{array}{l} \text{\%DFLE} \left\{ \begin{array}{l} \text{DVT:} \left\{ \begin{array}{l} \text{CR} \\ \text{MT} \\ \text{RD} \\ \text{MD} \\ \text{AD} \end{array} \right\} \left[ \text{VS:} \text{sssss} \right] \\ \\ \text{DV:} \left\{ \begin{array}{l} \text{CR} \\ \text{MT} \\ \text{MD} \\ \text{AD} \\ \text{RD} \end{array} \right\} \text{nr1} \end{array} \right. \\ \\ \left[ \text{,BEG:} \left[ \begin{array}{l} \left( \text{Znr4} \right) \\ \left( \left[ \text{Znr5} \right] \left\{ \text{,Bnr6} \right. \right. \right. \end{array} \right. \left. \left. \left. \left\{ \text{Aaa...a,} \right\}, \left\{ \text{Aaa...a} \right\}, \dots \right\} \right) \right] \right] \end{array} \right]$$

unde:

- BEG indică unitatea de imprimare cu care începe inventarierea și anume:
- cu zona indicată prin  $nr4$  ( $1 \leq nr4 \leq 99$ ), în cazul fișierelor nedefinite sau secvențiale (o zonă disc reprezintă un număr de cilindri succesivi din cadrul unui volum, iar o zonă bandă reprezintă o bobină de bandă);
- cu un anumit bloc sau sector din cadrul unei zone, în cazul fișierelor nedefinite sau secvențiale; numărul blocului sau sectorului este indicat prin  $nr6$  ( $1 \leq nr6 \leq 9999$ ), iar numărul zonei, prin  $nr5$  ( $1 \leq nr5 \leq 99$ );
- cu articolul ale cărui chei — specificate în ordinul KEY — au valorile definite alfanumeric (prin  $aa \dots a$ ) sau hexazecimal (prin  $xx \dots x$ ), în cazul fișierelor secvențiale sau secvențiale-indexate; numărul zonei în cadrul căreia se află articolul este indicat prin  $nr5$ ;



- cu primul articol din căsuța indicată prin  $nr7$  ( $1 \leq nr7 \leq 8.260.607$ ), în cazul fișierelor selective.

#### Observație

În absența argumentului BEG, imprimarea începe, funcție de modul de organizare a fișierului, cu primul bloc sau sector din cadrul primei zone, cu primul articol, sau cu primul articol din prima căsuță a fișierului.

**Ordinul DUMP** precizează modul de organizare a fișierului, forma de imprimare și, în cazul imprimării parțiale, condițiile de sfârșit ale inventarierii

*Format general:*

$$\% \text{ DUMP } \left[ \begin{array}{l} \text{UND, } \left\{ \begin{array}{l} \text{EOF} \\ \text{BLnr1} \end{array} \right\} \\ \left\{ \begin{array}{l} \text{SEQ} \\ \text{INS} \end{array} \right\}, \left\{ \begin{array}{l} \text{EOF} \\ \left( \left\{ \begin{array}{l} \text{Aaa} \dots \text{a} \\ \text{Xxx} \dots \text{x} \end{array} \right\}, \left\{ \begin{array}{l} \text{Aaa} \dots \text{a} \\ \text{Xxx} \dots \text{x} \end{array} \right\}, \dots \right) \\ \text{Rnr2} \end{array} \right\} \\ \text{RAN, } \left\{ \begin{array}{l} \text{EOF} \\ \text{Cnr3} \end{array} \right\} \end{array} \right] \left\{ \begin{array}{l} \text{A} \\ \text{X} \\ \text{AX} \end{array} \right\}$$

Modul de organizare a fișierului este indicat prin una din opțiunile UND (nedefinită), SEQ (secvențială), INS (secvențial-indexată) sau RAN (selectivă).

În cazul inventarierii totale imprimarea ia sfârșit în momentul detectării sfârșitului fișierului (opțiunea EOF).

În cazul inventarierii parțiale prelucrarea ia sfârșit după:

- imprimarea unui anumit număr ( $nr1$ ) de blocuri sau sectoare, în cazul în care organizarea fișierului este nedefinită ( $1 \leq nr1 \leq 99999$ );
- imprimarea articolului ale cărui chei-definite prin ordinul KEY-au valorile  $aa \dots a$  sau  $xx \dots x$ , în cazul în care organizarea fișierului este secvențial-indexată;
- imprimarea unui anumit număr ( $nr2$ ) de articole, în cazul în care organizarea fișierului este secvențială sau secvențial-indexată ( $1 \leq nr2 \leq 99999$ );

— imprimarea articolelor dintr-un anumit număr ( $nr3$ ) de căsuțe, în cazul în care organizarea fișierului este selectivă ( $1 \leq nr3 \leq 8.260.607$ ).

Forma de imprimare este precizată prin una din opțiunile: A (afli-numerică), X (hexazecimală) sau AX (mixtă).

**Ordinul KEY** precizează numărul, lungimea și poziția cheilor în cadrul articolului, fiind utilizat numai în cazul inventarierii parțiale.

*Format general:*

$$\% \text{ KEY } a_1, l_1, a_2, l_2, \dots, a_{12}, l_{12}$$

unde:

- $a_1, a_2, \dots, a_{12}$  indică adresele relative ale cheilor în raport cu începutul articolului ( $0 \leq a_i \leq 32767$ );
- $l_1, l_2, \dots, l_{12}$  indică lungimile cheilor ( $1 \leq l_i \leq 256$ ).



Numărul maxim de chei este 12, iar suma lungimilor zonelor ce le sînt asociate nu trebuie să depășească 256 locații.

Ordinul **END** marchează sfîrșitul parametrilor transmiși programului de inventariere.

*Format general:*

% END

#### Observație

În cadrul unei lucrări, succesiunea ordinelor transmise programului de inventariere, pentru realizarea operației de inventariere, este: DUMP, DELE, KEY și END.

Exemplele următoare ilustrează inventarierea totală a fișierului FURNIZORI și, respectiv, inventarierea parțială a fișierului secvențial COMENZI (inventarierea începe cu primul articol și se termină după imprimarea articolului care are valorile 40 și CTCE în zonele de adrese relative 20 și 40).

```
JOB INV1,AN:XXXX, PN:Y
SYSRUN INVFICH
% DUMP SEQ, X
% DFLE DVT:MT,VS: MT1MAN,FN:'FURNIZORI'
% END
EOJ
```

```
JOB INV2,AN:XXXX, PN:Y
SYSRUN INVFICH
% DUMP (XF4FO,ACTCE)
% DFLE DVT:AD,VS: AD1MAN,FN:'COMENZI'
% KEY 20, 2, 40, 4
% END
EOJ
```

## 2. REZUMAREA VOLUMELOR

*Operația de rezumare a unui volum* constă în tipărirea unei situații care conține, în principal, informații referitoare la etichetele asociate volumului și fișierelor înregistrate pe acest volum.

*Formatul rezumatului* este dependent de natura volumului.

● În cazul volumelor bandă organizate standard și care conțin un singur fișier, rezumatul — al cărui format este prezentat în figura XX.1 — conține următoarele informații:

- eticheta de identificare a volumului (VOL1);
- etichetele de început de fișier (HDR1 și HDR2);
- numărul înregistrărilor fizice (blocurilor) ce alcătuiesc fișierul (NB);
- etichetele de sfîrșit de fișier (EOF1 și EOF2);
- etichetele de sfîrșit de volum (EOV1 și EOV2).



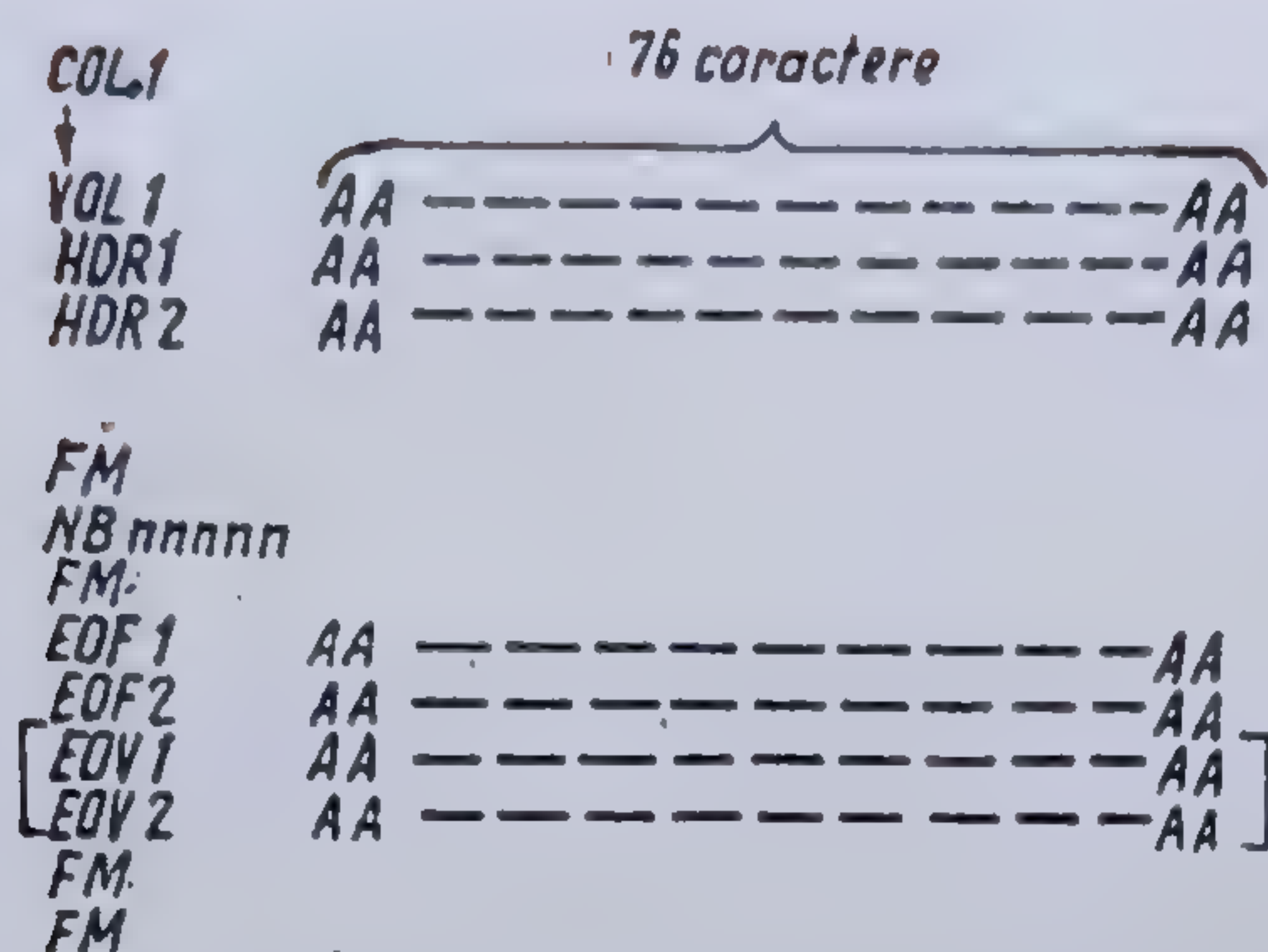


Fig. XX.1

- În cazul volumelor disc, rezumatul — al cărui format este prezentat în figura XX.2 — conține informațiile următoare:
- tabela de ocupare a volumului (TOV);
  - eticheta de identificare a volumului (VOL1);
  - etichetele asociate fișierelor înregistrate pe volum; pentru fiecare fișier se indică etichetele HDR1 și HDR2, iar în cazul fișierelor organizate

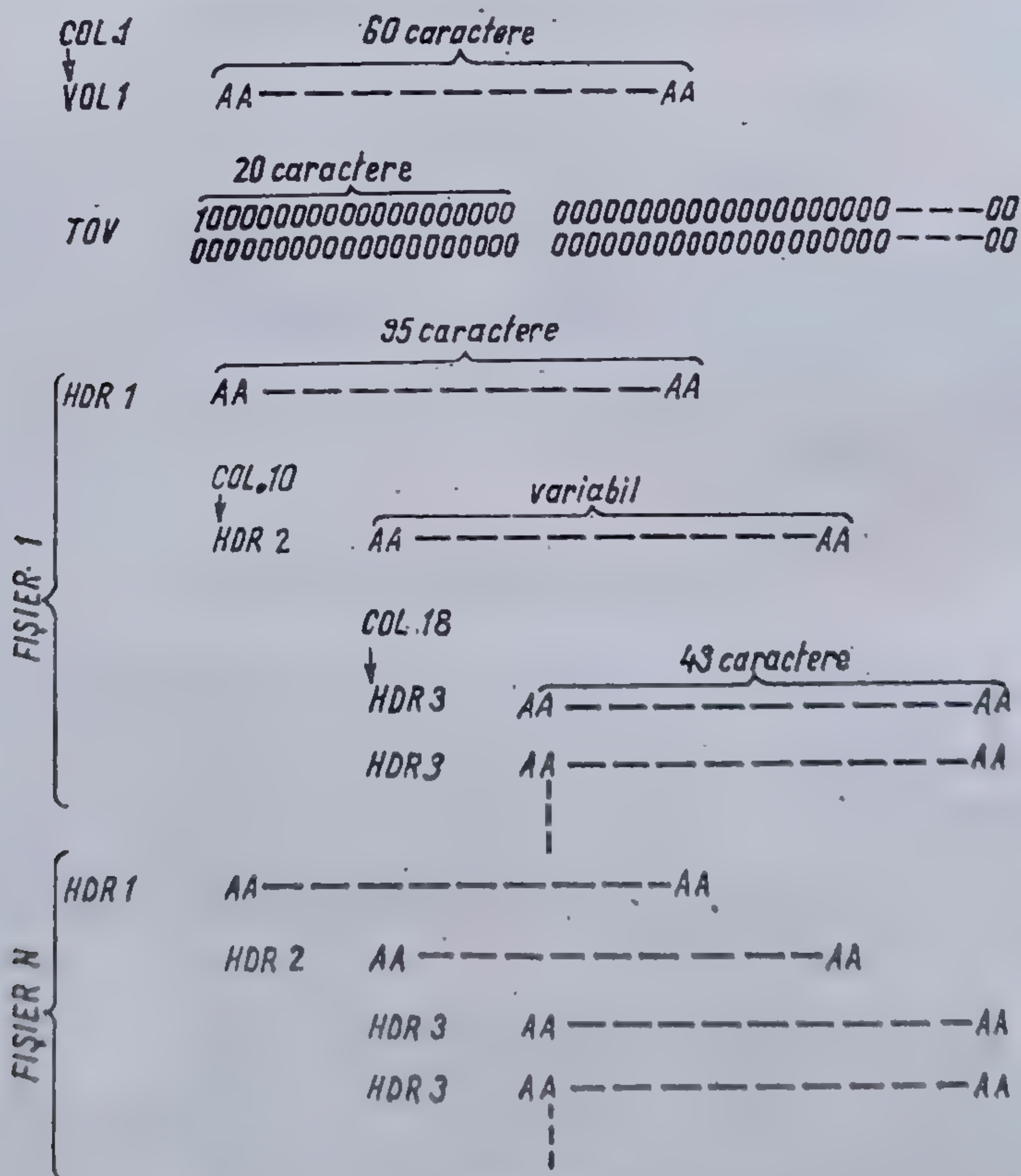


Fig. XX.2



secvențial-înlănțuit (în zonă partajată) se indică și etichetele HDR3 asociate fișierelor membre.

În tabela de ocupare a volumului, fiecărui cilindru al volumului îi corespunde un indicator care are valoarea 0, dacă cilindrul este disponibil, și valoarea 1, dacă cilindrul este alocat unui fișier.

Definirea operației de rezumare se realizează prin două ordine: RESUME și DVOL. Formatele acestor ordine sînt următoarele:

% RESUME

% DVOL	{	DV:	$\left\{ \begin{array}{c} MT \\ RD \\ MD \\ AD \end{array} \right\}$	VS:sssss	[ , ON: 'nume1']
			$\left\{ \begin{array}{c} MT \\ RD \\ MD \\ AD \end{array} \right\}$	nr1	

#### Observație

În cadrul unei lucrări, succesiunea ordinelor transmise programului de inventariere, pentru realizarea operației de rezumare este: RESUME, DVOL, END.

Exemplul următor prezintă ordinele pentru obținerea rezumatului unui volum disc și al unui volum bandă.

```

JOB REZ, AN:XXXX, PN: Y
SYSRUN INVFICH
% RESUME
% DVOL DVT:AD, VS:AD1MAN
% RESUME
% DVOL DVT: MT,VS:MT1MAN
% END
EOJ

```



## GENERATORUL DE PROGRAME DE SORTARE

## 1. GENERALITĂȚI

*Generatorul de programe de sortare* este un program utilitar — catalogat în BUS sub numele SMGEN — care, pe baza unor parametri transmiși de programator (parametri ce definesc o anumită operație de sortare), selectează din BSS modulele de sortare corespunzătoare (vezi subcapitolul 3 al capitolului XIX), generînd un program de sortare specific. Acest program, elaborat în format binar translatabil, este supus în continuare operației de editare a legăturilor, rezultînd un program executabil care poate fi catalogat într-o bibliotecă IMT — în vederea unei execuții ulterioare — sau executat imediat.

## 2. ORDINELE TRANSMISE GENERATORULUI

Informațiile transmise de programator generatorului de programe de sortare se referă, în principiu, la fișierele care intervin în prelucrare și la cheile de sortare. Aceste informații sînt furnizate prin intermediul unor ordine înregistrate pe cartele. Principalele ordine transmise generatorului sînt: GEN, INPUT, SCRATCH, OUTPUT și KEYS.

*Ordinul GEN* indică numărul căilor de sortare, metoda ce va fi utilizată în etapa de interclasare a monotoniilor, biblioteca IMT în care va fi eventual catalogat programul generat.

*Format general:*

-% GEN SORT  $\left\{ \begin{array}{l} \text{POLY} \\ \text{OSCT} \end{array} \right\}$ , WAY: nr1

$$\left[ \begin{array}{l} \text{,LN: nume1} \\ \text{,DVT: } \left\{ \begin{array}{l} \text{MT} \\ \text{RD} \\ \text{MD} \\ \text{AD} \end{array} \right\} \text{,VS: ssssss} \\ \text{,DV: } \left\{ \begin{array}{l} \text{MT} \\ \text{RD} \\ \text{MD} \\ \text{AD} \end{array} \right\} \text{nr2} \end{array} \right] \text{,GN:nr3, ,VN: nr4, FN:nume2}$$

unde:

- SORT este argumentul care indică generarea unui program de sortare;
- POLY și OSCT sînt argumentele care precizează metoda ce va fi utilizată în etapa de interclasare a monotoniilor și anume: metoda polifazelor (POLY) sau metoda oscilantă (OSCT);
- WAY indică numărul căilor de sortare ( $2 \leq \text{nr1} \leq 6$ ).



Celelalte argumente sînt utilizate în cazul în care programul de sortare generat va fi catalogat într-o bibliotecă IMT în vederea unei execuții ulterioare.

*Ordinul INPUT* precizează caracteristicile fișierului de intrare.

*Format general:*

$$\% \text{ INPUT } i, \text{ DVT: } \left\{ \begin{array}{c} \text{MT} \\ \text{RD} \\ \text{MD} \\ \text{AD} \end{array} \right\} \left[ , \text{RCF: } \left\{ \begin{array}{c} \text{FIX} \\ \text{VAR} \end{array} \right\} \right] , \text{RCS:}nr1 , \text{BFS:}nr2$$

unde:

—  $i$  reprezintă identificatorul de exploatare atribuit fișierului de intrare (o literă);

— RCF precizează formatul articolelor, care poate fi: fix (FIX) sau variabil (VAR);

— RCS indică dimensiunea maximă a articolelor ( $1 \leq nr1 \leq 65535$ );

— BFS indică dimensiunea maximă a înregistrării fizice — bloc sau pagină ( $1 \leq nr2 \leq 65535$ ).

*Ordinul SCRATCH* descrie caracteristicile fișierului de manevră.

*Format general:*

$$\text{SCRATCH } m[/n], \text{ DVT: } \left\{ \begin{array}{c} \text{MT} \\ \text{RD} \\ \text{MD} \\ \text{AD} \end{array} \right\}$$

unde:

—  $m$  reprezintă identificatorul de exploatare atribuit fișierului de manevră (o literă);

—  $n$  reprezintă identificatorul de exploatare atribuit fișierelor membre ale zonei partajate (o literă); acest argument este utilizat în cazul în care fișierul de manevră este organizat în zonă partajată.

*Observații*

— Dacă în etapa de interclasare a monotonilor este utilizată metoda polifazelor, pentru fișierul de manevră este necesar maximum de volum — în cazul în care fișierul este organizat în zonă partajată — sau  $N + 1$  volume disc sau bandă ( $N$  reprezintă numărul căilor de sortare).

— Dacă în etapa de interclasare a monotonilor este utilizată metoda oscilantă, fișierul de manevră poate ocupa, funcție de mărimea fișierului de intrare, o zonă disc, un volum sau mai multe volume disc.

*Ordinul OUTPUT* precizează caracteristicile fișierului de ieșire.

*Format general:*

$$\% \text{ OUTPUT } o, \text{ DVT: } \left\{ \begin{array}{c} \text{MT} \\ \text{RD} \\ \text{MD} \\ \text{AD} \end{array} \right\} \left[ , \text{RCF: } \left\{ \begin{array}{c} \text{FIX} \\ \text{VAR} \end{array} \right\} \right] \text{RCS:}nr1, \text{BFS:}nr2$$



unde:

—  $o$  reprezintă identificatorul de exploatare atribuit fișierului de ieșire (o literă).

Semnificația celorlalte argumente ale ordinului OUTPUT este aceeași ca cea a argumentelor ordinului INPUT.

Ordinul **KEYS** definește cheile de sortare.

Format general:

$$\% \text{KEYS } K1: \left( l_1 a_1, \begin{Bmatrix} \text{ASC} \\ \text{DSC} \end{Bmatrix}, \begin{Bmatrix} \text{ALN} \\ \text{DEC} \\ \text{DPK} \\ \text{BIN} \\ \text{FLT} \end{Bmatrix} \right), \dots, K12: \left( l_{12} a_{12}, \begin{Bmatrix} \text{ASC} \\ \text{DSC} \end{Bmatrix}, \begin{Bmatrix} \text{ALN} \\ \text{DEC} \\ \text{DPK} \\ \text{BIN} \\ \text{FLT} \end{Bmatrix} \right)$$

unde:

—  $k_1, \dots, k_{12}$  indică prioritatea cheilor în ordine descrescătoare,

—  $a_1, \dots, a_{12}$  indică adresele relative ale zonelor asociate cheilor, în cadrul zonei-articol ( $0 \leq a_i \leq 65535$ );

—  $l_1, \dots, l_{12}$  indică lungimea zonelor asociate cheilor ( $1 \leq l_i \leq 256$  locații);

— opțiunile ASC și DSC indică sensul sortării: crescător și, respectiv, descrescător;

— opțiunile ALN, DEC, DPK, BIN și FLT indică formatul de reprezentare a valorilor cheilor: în cod EBCDIC, în zecimal despachetat, în zecimal împachetat, în cod complementar și, respectiv, în virgulă mobilă.

Ordinul **END** marchează sfârșitul parametrilor transmiși generatorului.

Format general:

**% END**

#### Observații

— Pe baza parametrilor furnizați prin ordinele prezentate, generatorul de programe de sortare generează un program de sortare specific, care nu poate fi însă executat, deoarece nu sînt cunoscute informațiile de identificare a fișierelor (de intrare, ieșire și de manevră) și informațiile privind unitățile periferice ce trebuie afectate acestor fișiere. Aceste informații pot fi precizate prin cartelele de comandă LABEL și ASSIGN, sau prin ordinele LABEL și ASSIGN specifice generatorului (acestea conțin aceleași argumente ca și cartelele de comandă corespundente).

— Primul ordin transmis generatorului trebuie să fie GEN, iar ultimul, END; poziția celorlalte ordine în cadrul secvenței delimitate prin ordinele GEN și END poate fi oarecare.

Două exemple (programele XXI.1 și XXI.2) sînt prezentate în continuare pentru ilustrarea utilizării generatorului de programe de sortare.

Primul exemplu realizează sortarea articolelor fișierului RETETE, creat în programul XVIII.3, crescător după valorile datei COD-PRODUS. Deoarece în etapa de interclasare a monotoniilor este utilizată metoda polifazelor, pentru fișierul de manevră s-au afectat — corespunzător numărului căilor de sortare — trei volume de bandă.

În cel de al doilea exemplu sînt sortate, prin metoda oscilantă, articolele aceluiași fișier.

Cheile de sortare sînt datele COD-PRODUS și DENUMIRE-PRODUS.



```

*          JOB SORT1,AN:IS00,PN:ANS
*          INIT DVT:MT,VS:MT1MAN
*          INIT DVT:MT,VS:MT2MAN
*          INIT DVT:MT,VS:MT3MAN
*          SYSRUN SMGEN
% GEN SORT POLY,WAY:2
% INPUT A,DVT:AD,RCF:VAR,RCS:273,BFS:4095
% ASSIGN A,DVT:AD,VS:AD1MAN
% LABEL A,FN:'RETETE'
% SCRATCH B,DVT:MT
% ASSIGN B,DVT:MT,VS:MT1MAN+MT2MAN+MT3MAN
% LABEL B,FN:'MANEVRA'
% OUTPUT C,DVT:AD,RCF:VAR,RCS:273,BFS:4095
% ASSIGN C,DVT:AD,VS:AD1MAN
% LABEL C,FN:'RETETES',AM:ANY,SZ:5
% KEYS K1:(6,1,ASC)
% END

```

```

*
RUN
EOJ

```

PROGRAMUL XXI.1,

```

*          JOB SORT2,AN:IS00,PN:ANS
*          INIT DVT:MT,VS:MT2MAN
*          SYSRUN SMGEN
% GEN SORT DSCT,WAY:4
% INPUT A,DVT:AD,RCF:VAR,RCS:273,BFS:4095
% ASSIGN A,DVT:AD,VS:AD1MAN
% LABEL A,FN:'RETETE'
% SCRATCH B,DVT:AD
% ASSIGN B,DVT:AD,VS:AD1MAN
% LABEL B,FN:'MANEVRA',AM:ANY,SZ:5
% OUTPUT C,DVT:MT,RCF:VAR,RCS:273,BFS:2000
% ASSIGN C,DVT:MT,VS:MT2MAN
% LABEL C,FN:'RETETES'
% KEYS K1:(6,1,ASC),K2:(20,7,ASC)
% END

```

```

*
RUN
EOJ

```

PROGRAMUL XXI.2



## CAPITOLUL XXII

### PROGRAMUL DE ÎNTREȚINERE A FIȘIERELOR SI VOLUMELOR

În practică apare frecvent necesitatea efectuării unor operații de întreținere și control al datelor înregistrate și păstrate pe suport magnetic. Sistemele de operare ale calculatoarelor de tipul FELIX C, asigură definirea și realizarea acestor operații, prin intermediul programului utilitar de întreținere a fișierelor și volumelor, program catalogat în BUS sub numele MAINT.

#### 1. FUNCȚIILE PROGRAMULUI

Unitatea de prelucrare pentru programul de întreținere este volumul (rola de bandă, pachetul de discuri) sau fișierul (organizarea acestuia putând fi: nedefinită, secvențială, secvențial-indexată sau selectivă), iar principalele funcții realizate se referă la:

- copierea fișierelor și volumelor;
- salvarea fișierelor și volumelor;
- restaurarea fișierelor și volumelor;
- reorganizarea fișierelor secvențial-indexate;
- reorganizarea fișierelor selective.

#### 2. ORDINELE ADRESATE PROGRAMULUI DE ÎNTREȚINERE

Există două categorii de ordine interpretate de programul de întreținere:

- cele care indică funcțiile solicitate;
- cele care descriu aceste funcții.

Aceste ordine sînt înregistrate pe cartele, conform sintaxei specifice programelor utilitare.

a. Descrierea fișierelor și volumelor. În prelucrările efectuate de programul de întreținere intervin două fișiere, două volume sau un fișier și un volum. Aceste unități de prelucrare sînt definite prin patru ordine: OLDFLE, NEWFLE, OLDVOL, NEWVOL.

Format general:

$$\% \left\{ \begin{array}{l} \text{OLDFLE} \\ \text{NEWFLE} \end{array} \left[ \text{AM:} \left\{ \begin{array}{l} \text{ANY} \\ \text{NEW} \end{array} \right\} \right] [\text{SZ:}nr1] \right\} \left( \begin{array}{l} \text{DV:} \left\{ \begin{array}{l} \text{RD} \\ \text{AD} \\ \text{MD} \\ \text{MT} \end{array} \right\} nr2 \\ \text{DVT:} \left\{ \begin{array}{l} \text{RD} \\ \text{AD} \\ \text{MD} \\ \text{MT} \end{array} \right\}, \text{VS:}ssssss \end{array} \right)$$



FN:numel [,GN:nr3] [,VN:nr4] [,ON:nume2]

$$\% \left\{ \begin{array}{l} \text{OLDVOL} \\ \text{NEWVOL} \end{array} \right\} \left\{ \begin{array}{l} \text{DV:} \left\{ \begin{array}{l} \text{RD} \\ \text{AD} \\ \text{MD} \\ \text{MT} \end{array} \right\} \text{nr1} \\ \text{DVT:} \left\{ \begin{array}{l} \text{RD} \\ \text{AD} \\ \text{MD} \\ \text{MT} \end{array} \right\} \text{,VS:ssssss} \end{array} \right\} [,ON:numel]$$

b. **Descrierea operațiilor.** Copierea unui fișier sau volum se indică prin ordinul COPY:

$\% \text{ COPY } \left\{ \begin{array}{l} \text{VOL} \\ \text{FLE,} \end{array} \right\} \left\{ \begin{array}{l} \text{SEO} \\ \text{UND} \end{array} \right\}$

Prin copiere unui fișier sau unui volum i se schimbă (convertește) suportul de exploatare.

Copierea unui fișier secvențial nu este restricționată de tipul suportului (adică se pot copia fișiere de pe bandă pe disc și invers). În schimb, celelalte tipuri de fișiere (nedefinite, secvențial-indexate și selective) nu pot fi copiate decât pe suporturi de același tip.

#### Observație

Fișierele secvențial-indexate și selective sînt considerate de către programul de întreținere ca fiind de organizare nedefinită.

**Salvarea** este o operație care are sens numai pentru volumele disc sau fișierele nedefinite avînd ca suport discul. Ea constă în stocarea temporară a conținutului acestora pe bandă magnetică, sub forma unui fișier de organizare nedefinită (care nu poate fi exploatat direct). Prin această operație se evită utilizarea pachetelor de discuri pentru păstrarea informațiilor.

Ordinul prin care se indică realizarea unei operații de salvare este SAVE:

$\% \text{ SAVE } \left\{ \begin{array}{l} \text{VOL} \\ \text{FLE} \end{array} \right\}$

**Restaurarea** este operația inversă salvării.

Constă în înregistrarea pe volumul inițial, în scopul exploatării, a informațiilor ce au făcut obiectul unei salvări și se indică prin ordinul REST:

$\% \text{ REST } \left\{ \begin{array}{l} \text{VOL} \\ \text{FLE} \end{array} \right\}$

**Reorganizarea** unui fișier secvențial-indexat constă în rescrierea acestuia într-o zonă distinctă, pe același volum sau pe un alt volum, eliminîndu-se partea de depășire și articolele invalidate (șterse) din partea principală. Se indică prin ordinul ISORG:

$\% \text{ ISORG}$



Reorganizarea unui fișier selectiv constă în eliminarea articolelor invalidate din partea de depășire, regruparea articolelor citite mai frecvent din partea de depășire în partea principală (pentru a efectua această operație programul de întreținere are în vedere valorile contorului de actualizare asociat fiecărui articol) și rescrierea fișierului obținut într-o zonă distinctă, pe același volum sau pe un alt volum.

Operația de reorganizare se indică prin *ordinul* RDORG:

% RDORG DLC[,LIST]

unde:

DLC — indică poziționarea contorului de actualizare la valoarea zero;

LIST — indică listarea numărului de articole din fiecare casuță

Exemplele următoare ilustrează descrierea operațiilor prezentate.

```

.      JOB M1,AN:1SOO,PN:FELIX
.      *
.      * COPIERE FISIER
.      *
.      INIT DVT:AD,VS:AD1MAN
.      SYSRUN MAINT
% COPY FLE
% OLDFLE DVT:MT,VS:MT1MAN,FN:'FISBAND'
% NEWFLE DVT:AD,VS:AD1MAN,FN:'FISDISC',AM:ANY,SZ:10
% END
      EOJ

.      JOB M2,AN:1SOO,PN:FELIX
.      *
.      * SALVARE VOLUM
.      *
.      INIT DVT:MT,VS:MT1MAN
.      SYSRUN MAINT
% SAVE VOL
% OLDFLE DVT:AD,VS:AD1MAN
% NEWFLE DVT:MT,VS:MT1MAN,FN:'FISSALVAT'
% END
      EOJ

.      JOB M3,AN:1SOO,PN:FELIX
.      *
.      * RESTAURARE VOLUM
.      *
.      INIT DVT:AD,VS:AD1MAN
.      SYSRUN MAINT
% REST VOL
% OLDFLE DVT:MT,VS:MT1MAN,FN:'FISSALVAT'
% NEWVOL DVT:AD,VS:AD1MAN
% END
      EOJ

```



JOB M4,AN:ISOO,PN:FELIX

\*

\* REORGANIZARE FISIER SECV-INDEXAT SI  
SELECTIV

\*

SYSRUN MAINT

% ISORG

% OLDFLE DVT:AD,VS:AD1MAN,FN:'SECVIND',GN:1,VN:0

% NEWFLE DVT:AD,VS:AD1MAN,FN:'SECVIND',GN:2,VN:0,  
,AM:ANY,SZ:15

% RDORG DLC,LIST

% OLDFLE DVT:AD,VS:AD2MAN,FN:'SELECTIV'

% NEWFLE DVT:AD,VS:AD1MAN,FN:'SELECTIV',AM:ANY,SZ:20,GN2

% END

EOJ



## CAPITOLUL XXIII

### UTILIZAREA BIBLIOTECILOR

#### 1. GENERALITĂȚI

O aplicație de prelucrare automată a datelor conține, în general, un număr mare de programe a căror manevrare — în fazele de testare și exploatare — devine dificilă atunci când suportul de înregistrare a acestor programe este cartela.

Înlăturarea acestui neajuns se poate realiza prin înregistrarea acestor programe pe suporturi magnetice, în cadrul unor colecții organizate numite *biblioteci*, care pot fi gestionate prin intermediul unui program specializat al sistemului de operare, numit *bibliotecar*.

Biblioteca reprezintă o colecție de programe de același format, numite *cărți*. Corespunzător formatelor programelor există trei tipuri de biblioteci:

- *bibliotecă sursă*, în care o carte reprezintă un program sursă sau o secvență a unui program sursă;

- *biblioteci BT*, în care o carte reprezintă rezultatul unei singure execuții a unui program de traducere;

- *biblioteci IMT*, în care o carte reprezintă un program executabil (IMT), alcătuit din unul sau mai multe segmente.

Suportul bibliotecilor poate fi discul (suportul principal) sau banda magnetică.

Din punctul de vedere al SGF-ului — prin intermediul căruia se realizează operațiile de intrare/ieșire necesare gestionării bibliotecilor — biblioteca reprezintă un *multifișier*, iar cărțile bibliotecii, fișiere membre ale acestuia.

Biblioteca este identificată prin *nume* (succesiune de maximum 8 caractere, primul fiind o literă), *număr de generație* și *număr de versiune*.

Cărțile unei biblioteci sînt identificate prin *nume* și *număr de actualizare*. Numele cărții reprezintă o succesiune de maximum 8 caractere, primul fiind o literă, iar numărul de actualizare (care permite identificarea a două sau a mai multor cărți care se află în aceeași bibliotecă și au același nume) poate lua valori între 1 și 255.

În cazul bibliotecilor organizate pe disc (care fac obiectul prezentării ce urmează), pentru regăsirea rapidă a cărților unei biblioteci, aceasta conține un *fișier repertoriu* (care este unul din fișierele membre ale multifișierului) în care sînt înregistrate, pentru fiecare carte numele, numărul de actualizare și adresa de început a cărții.

#### 2. BIBLIOTECARUL

a. **Funcțiile bibliotecarului.** *Principalele operații* realizate de bibliotecar (program utilitar catalogat în BUS sub numele BIBLIO) sînt următoarele:

- crearea unei biblioteci;
- actualizarea unei biblioteci;



- imprimarea repertoriului unei biblioteci;
- imprimarea unei cărți.

Definirea acestor operații se realizează prin intermediul unui limbaj de control, propriu bibliotecarului. Ordinele acestui limbaj sînt structurate pe patru niveluri (tabelul XXIII.1).

TABELUL XXIII.1

Nivel	Ordin	Funcția
1	OPNLIB	Deschiderea bibliotecii
2	INCLUDE	Inserarea unei cărți
	REPLACE	Înlocuirea unei cărți
	DELETE	Ștergerea unei cărți
	EDIT	Modificarea unei cărți sursă
	REPRT	Imprimarea repertoriului
	DISPLAY	Imprimarea unei cărți
3	MOD	Modificarea unei cărți sursă
4	ENDLIB	Închiderea bibliotecii

La alcătuirea programelor de control adresate bibliotecarului trebuie să se țină seama de faptul că unui ordin de nivel  $n$  ( $n = 1, 2, 3$ ) trebuie să-i urmeze un ordin de nivel  $n + 1$ .

b. Crearea unei biblioteci constă în inserarea (catalogarea) uneia sau mai multor cărți, în zona partajată afectată bibliotecii. Cărțile se pot afla într-o altă bibliotecă (de același tip) sau în fișierul de lucrări, perforate pe cartele.

• Operația de premarcare a zonei destinată a primi cărțile este indicată de programator în cartela ALLOC, a cărei interpretare și execuție sînt realizate de moniton. Argumentele care trebuie specificate în această cartelă sînt cunoscute. O particularitate o prezintă argumentul FN a cărei sintaxă este următoarea:

FN:  $\text{'nume' } \left\{ \begin{array}{l} \text{SOU} \\ \text{RBN} \\ \text{IMT} \end{array} \right\},$

unde:

- *nume* reprezintă numele bibliotecii;
- SOU, RBN, IMT sînt opțiuni care indică tipul bibliotecii: sursă, BT, IMT.

#### Observație

Dacă numărul caracterelor care alcătuiesc numele bibliotecii este mai mic decît 8, diferența pînă la 8 se completează cu spații.



● Operația de inserare a unei cărți într-o bibliotecă (ca de altfel și celelalte operații efectuate asupra bibliotecii sau asupra cărților care o compun) trebuie să fie precedată de deschiderea bibliotecii, care constă în verificarea de către bibliotecar, a parametrilor de identificare a bibliotecii.

Acești parametrii sînt indicați de programator în *ordinul OPNLIB*:

$$\% \text{ OPNLIB } i, \text{LN:} \textit{nume}, \left\{ \begin{array}{l} \text{DVT: } \left\{ \begin{array}{l} \text{RD} \\ \text{MD} \\ \text{AD} \end{array} \right\}, \text{VS:} \textit{ssssss} \\ \text{DV: } \left\{ \begin{array}{l} \text{RD} \\ \text{MD} \\ \text{AD} \end{array} \right\} \textit{nr1} \end{array} \right\}$$

$$[\text{GN:} \textit{nr2}] [\text{VN:} \textit{nr3}], \text{FT: } \left\{ \begin{array}{l} \text{SOU} \\ \text{RBN} \\ \text{IMT} \end{array} \right\} [\text{INIT}]$$

unde:

— *i* reprezintă identificatorul de exploatare a bibliotecii; este o literă (asociată bibliotecii în momentul deschiderii) care trebuie să apară în toate ordinele ce definesc operații asupra bibliotecii sau componentelor sale;

— *FT* precizează tipul bibliotecii;

— *INIT* este argumentul care trebuie specificat cînd se inserează prima carte în bibliotecă, indicînd bibliotecarului faptul că trebuie să inițializeze zona ce va conține repertoriul bibliotecii.

● Operația de catalogare a unei cărți este definită prin *ordinul INCLUDE*:

*% INCLUDE OL:o, OF:numeo, [ON:nro], DL:d*

unde:

— *OL* indică identificatorul de exploatare al bibliotecii sau al fișierului de lucrări care conțin cartea ce va fi catalogată; identificatorul de exploatare al bibliotecii este o literă, iar al fișierului de lucrări, \*1;

— *OF* și *ON* precizează numele și numărul de actualizare ale cărții care va fi catalogată;

— *DL* indică identificatorul de exploatare a bibliotecii în care este inserată cartea.

#### Observație

Semnificația primei litere din cuvintele cheie *OL*, *OF*, *ON* și *DL* este: *O* — origine  
*D* — destinație.

— Dacă în *ordinul INCLUDE* se indică operația de catalogare a unei cărți care se află în fișierul de lucrări, cartelele ce compun cartea trebuie să urmeze imediat acestui ordin.

— Numărul de actualizare atribuit cărții catalogate este stabilit de bibliotecar astfel:

— dacă argumentul *ON* este utilizat numărul de actualizare va fi egal cu *nro*;

— dacă argumentul *ON* lipsește numărul de actualizare atribuit este 1, dacă în bibliotecă nu mai există cărți cu același nume; în caz contrar, numărul de actualizare se stabilește adăugînd 1 la cel mai mare dintre numerele de actualizare asociate cărților cu același nume.

— La catalogarea unei cărți în format sursă bibliotecarul numerotează liniile sursă începînd cu valoarea inițială 1 și rația 1.



```

•      JOB R1,AN:IS00,FN:ANS
•      INIT DVT:AD,VS:ADIMAN
•      ALLOC DVT:AD,VS:ADIMAN,FN:'SURSA' SOU',GN:1,VN:1,AN:ANY,SZ:5
•      SYSRUN BIBLIO
% OPNLIB A,DVT:AD,VS:ADIMAN,LN:SURSA,GN:1,VN:1,FT:SOU,INIT
% INCLUDE OL:*1,OF:CARTE1,DL:A
  ID DIVISION.
  PROGRAM-ID. CARTE1.
  ENVIRONMENT DIVISION.
  CONFIGURATION SECTION.
  DATA DIVISION.
  WORKING-STORAGE SECTION.
  01 CARTELA.
    05 A PIC 99.
    05 B PIC 99.
    05 C PIC 99.
  01 REZULTATE.
    05 R1 PIC 999.
  PROCEDURE DIVISION.
  START.
    ACCEPT CARTELA
    COMPUTE R1 = A + B - C
    DISPLAY 'R1= ' R1
    STOP RUN.
% INCLUDE OL:*1,OF:CARTE2,DL:A
  READ(105,1)N
  1* FORMAT(I3)
  IS=0
  DO 2 I=1,N
    IS=IS+1
  2 CONTINUE
  WRITE(108,3)N,IS
  3 FORMAT(10X,'SUMA PRIMELOR ',I4,' NEMERE NATURALE ESTE ',I10)
  STOP
  END
% ENDLIB
      EOJ

```

#### PROGRAMUL XXIII.1

● Sfârșitul unui program de control adresat bibliotecarului este marcat prin *ordinul* **ENDLIB**:

**% ENDLIB**

Programul XXIII.1 exemplifică operația de creare a bibliotecii sursă SURSA, în care sînt catalogate două cărți — CARTE1 și CARTE2.

c. Actualizarea unei biblioteci constă în adăugarea de noi cărți, înlocuirea, ștergerea sau modificarea unor cărți.

● Operația de adăugare se realizează prin *ordinul* **INCLUDE**.

● Operația de înlocuire a unei cărți dintr-o bibliotecă cu o altă carte, aflată într-o altă bibliotecă sau în fișierul de lucrări, este descrisă prin *ordinul* **REPLACE**:

**% REPLACE DL: d, DF: numed, [DN: nrd], OL: o, [ON: nro]**

unde:

— **DL**, **DF**, și **DN** indică identificatorul de exploatare al bibliotecii care conține cartea ce va fi înlocuită, numele și numărul de actualizare ale acestei cărți:

— **OL** și **ON** indică identificatorul de exploatare al bibliotecii sau al fișierului de lucrări care conține noua carte și numărul de actualizare al acestei cărți.



## Observație

Dacă noua carte se află în fișierul de lucrări, cartelele care o compun trebuie să urmeze ordinul REPLACE.

● Operația de ștergere a unei cărți dintr-o bibliotecă este definită prin **ordinul DELETE**:

% DELETE DL: *d*, DF: *numed* [,DN: *nrđ*]

● Efectuarea de modificări asupra unor cărți este posibilă în oricare din cele trei formate ale cărților (sursă, BT și IMT) și se realizează prin intermediul unor ordine specifice. În cele ce urmează vor fi prezentate **ordinele EDIT** și **MOD** care definesc operația de modificare a cărților în format sursă.

*Formatul ordinului EDIT* este următorul:

% EDIT OL: *o*, OF: *numeo* [,ON: *nro*] [', {DEL}'] [', {NLST}']  
[', {RET}'] [', {LIST}']

unde:

— *DEL/RET* indică bibliotecarului faptul că vechea carte trebuie ștersă/reținută;

— *NLST/LIST* validează/invalidază tipărirea la imprimantă a noii cărți, obținută în urma operației de modificare.

*Ordinul MOD* are sintaxa următoare:

% MOD *nr1* [,*nr2*]

Operația de modificare a unei cărți sursă constă în:

— adăugarea (inserarea) unor linii sursă; în acest caz este utilizat numai argumentul *nr1* care indică numărul de secvență al liniei sursă după care se va efectua adăugarea;

— ștergerea unor linii sursă succesive; în acest caz este utilizat atât argumentul *nr1*, care indică numărul de secvență al primei linii sursă ce va fi ștersă, cât și argumentul *nr2*, care indică numărul de secvență al ultimei linii sursă ce va fi ștersă;

— ștergerea unor linii sursă succesive și înlocuirea lor prin alte linii sursă; în acest caz sînt utilizate argumentele *nr1* și *nr2*, iar semnificația lor este cea prezentată anterior.

Liniile sursă adăugate sau cele care înlocuiesc o secvență trebuie să urmeze imediat ordinului MOD.

După efectuarea modificărilor specificate în ordinul (ordinele) MOD, bibliotecarul numerotează liniile sursă ale noii cărți începînd cu valoarea 1 și rația 1.

d. **Imprimarea repertoriului.** *Repertoriul unei biblioteci* este o situație imprimată care conține următoarele informații: numele, formatul, generația și versiunea bibliotecii; numele și numerele de actualizare ale cărților.

Operația de imprimare a repertoriului este definită prin **ordinul REPERT**:

% REPERT OL: *o*

e. **Imprimarea unei cărți** este descrisă prin **ordinul DISPLAY**:

% DISPLAY OL: *o*, OF: *numeo* [,ON: *nro*]



```

JOB R2,AN:IS00,PN:ANS
SYSRUN BIBLIO
% OPNLIB A,DVT:AD,VS:AD1HAN,LN:SURSA,GN:1,VN:1,FT:SOU
% INCLUDE OL:*1,OF:CARTE3,DL:A
FD:'FIS-PERSONAL-BANDA-1 RECORDING F LABEL RECORD STANDARD
BLOCK 10-RECORDS.
% REPLACE OL:*1,DF:CARTE2,DL:A
01. ART-C.
05 NUMAR-MATRICOL PIC 9(7).
05 NUME PIC A(20).
05 ADRESA PIC X(20).
05 DATA-NASTERII PIC 9(6).
05 SEX PIC A.
05 VIRSTA PIC 99.
05 FUNCTIE PIC A(12).
05 RETRIBUTIE PIC 9(5).
% EDIT OL:A,OF:CARTE1,RET
% MOD 12 05. R2 PIC 9(9).
% MOD 17 COMPUTE R2 = A * B
DISPLAY 'R2= ' R2'
% DELETE OL:A,DF:CARTE1,DN:1
% DISPLAY OL:A,OF:CARTE1,ON:2.
% REPERT OL:A
% ENDLIB.
EOJ

```

#### PROGRAMUL XXIII.2

Forma de imprimare este alfanumerică, pentru cărțile de format sursă, și hexazecimală, pentru cărțile de format BT sau IMT.

Programul XXIII.2 exemplifică operația de actualizare a bibliotecii SURSA și anume:

— adăugarea cărții CARTE3, care reprezintă o rubrică de descriere a unui fișier;

— înlocuirea cărții CARTE2 printr-o carte care reprezintă ansamblul rubricilor de descriere a datelor unui articol;

— modificarea cărții CARTE1 — care realizează calculul expresiei  $R1 = A + B - C$  și tipărirea rezultatului R1 — astfel încât noua carte (care va primi același nume — CARTE1) să realizeze în plus calculul expresiei  $R2 = A \times B$  și tipărirea rezultatului R2;

— ștergerea cărții CARTE1, cu numărul de actualizare 1;

— imprimarea cărții CARTE1, cu număr de actualizare 2;

— imprimarea repertoriului bibliotecii SURSA.

#### Observații

● Catalogarea textelor sursă în bibliotecă sursă poate fi realizată numai prin intermediul bibliotecarului.

Apelarea acestor cărți, în vederea compilării, se realizează:

— prin instrucțiunea COPY — în cazul textelor COBOL — al cărui format este:

$$COPY' \left\{ \begin{array}{l} DVT: \left\{ \begin{array}{l} RD \\ MD \\ AD \end{array} \right\}, VS: ssssss \\ DVT: \left\{ \begin{array}{l} RD \\ MD \\ AD \end{array} \right\}, nr1 \end{array} \right\}, FN: 'numel\ SOU/numo\ 2', GN: nr2, VN: nr3.$$

UN: nr4.



— prin *directiva* **FETCHS** — în cazul textelor **FORTRAN** — al cărei format este:

$$\bullet \text{ FETCHS } \left\{ \begin{array}{l} \text{DVT: } \left\{ \begin{array}{l} \text{RD} \\ \text{MD} \\ \text{AD} \end{array} \right\}, \text{VS:}ssssss \\ \text{DV: } \left\{ \begin{array}{l} \text{RD} \\ \text{MD} \\ \text{AD} \end{array} \right\}, nr1 \end{array} \right\}, \text{FN: } numel, \text{GN: } nr2, \text{VN: } nr3, \text{UN: } nr4, \text{LN: } nume2$$

unde *numel* reprezintă numele bibliotecii, *nume2*, numele cărții, iar *nr4*, numărul de actualizare al cărții.

În cazul în care textul sursă apelat este un program complet, instrucțiunea **COPY** și *directiva* **FETCHS** trebuie să urmeze imediat cartelei **COMPILE**.

● Catalogarea textelor (modulelor) obiect în biblioteci **BT** poate fi realizată prin intermediul bibliotecarului sau al compilatorului.

Apelarea cărților în vederea editării legăturilor se realizează prin *cartela* **FETCHB**. Apelarea cărților provenite din compilarea programelor **COBOL** prezintă următoarea particularitate:

— în *cartela* **FETCHB** trebuie indicat numele atribuit segmentului de program de către compilator la catalogarea acestuia în bibliotecă (nume anunțat printr-un mesaj la imprimantă), iar *cartela* **FETCHB** trebuie să fie precedată de o *cartelă* **SEG** în care se indică numele atribuit segmentului de program de către compilator (vezi capitolul **XIX**, subcapitolul 2).

● Catalogarea textelor (segmentelor) **IMT** în biblioteci **IMT** poate fi realizată prin intermediul bibliotecarului sau al editorului de legături.

Apelarea cărților în vederea execuției se realizează prin *cartela* **FETCH**.



## CAPITOLUL XXIV

### UTILIZAREA PROCEDURILOR CATALOGATE

Funcțiunile produselor informatice mai ample sînt realizate, de regulă, prin execuția unor combinații de mai multe programe. În scopul fixării acestor combinații sînt utilizate, frecvent, procedurile catalogate. Ele rezolvă într-o manieră simplă necesitățile de grupare a programelor în diferite variante de execuție, asigurînd, prin parametrizare, independența acestora față de condițiile externe care se pot schimba de la o execuție la alta.

#### 1. CREAREA PROCEDURILOR CATALOGATE

O procedură catalogată reprezintă o secvență de cartele de comandă și, eventual, de date, catalogată într-o bibliotecă de tip sursă. Catalogarea se realizează prin intermediul programului bibliotecar, într-o bibliotecă sursă utilizator sau a sistemului de operare (biblioteca sistem Z%PROC).

În secvența de cartele ce alcătuiesc o procedură catalogată nu trebuie să figureze cartelele JOB, EOJ și EOF. De asemenea, aceste cartele, fiind tratate de către programul bibliotecar, nu trebuie să conțină caracterul % în prima coloană.

Textul unei proceduri catalogate poate conține zone substituabile, delimitate, la început și la sfîrșit, prin caracterul &. În momentul creării procedurii aceste zone au o valoare inițială, care poate fi sau nu modificată atunci cînd procedura este apelată pentru execuție. Din acest motiv, zonele substituabile se mai numesc și parametri formali. Valoarea unui parametru formal reprezintă un șir de caractere alfanumerice (cu excepția caracterelor \_ și :), care nu pot fi înregistrate decît pe o singură cartelă.

Exemplul următor (programul XXIV.1) prezintă catalogarea, în bibliotecă sursă PROCAT, a două proceduri: CATAL1 și CATAL2.

Prima procedură — CATAL1 — conține cartele de comandă pentru apelarea pentru execuție, din biblioteca BIBIMT, a două programe IMT, CREARE și CONSULT, care realizează crearea și, respectiv, consultarea fișierului pe bandă CONTRACTE. Parametrii substituabili SD și SB, cu valorile inițiale AD1MAN și MT1MAN, au fost stabiliți astfel încît să se asigure independența execuției celor două programe de unitățile periferice disponibile.

A doua procedură — CATAL2 — conține cartele de comandă necesare apelării programului utilitar de întreținere a fișierelor și descrierii unei operații ce poate fi realizată de acesta. Dar, codul acestei operații nu este precizat, parametrul OPERATIE neprimind o valoare inițială (aceasta va fi precizată la apelarea pentru execuție a procedurii). Procedura CATAL2 ilustrează și maniera în care pot fi inserate, în textul procedurilor, cartele care încep cu caracterul %.



```

*      JOB B3,AN:IS00,PN:FELIX
*      INIT DVT:AD,VS:AD1MAN
*      ALLOC DVT:AD,VS:AD1MAN,FN:'PROCAT SOU',AM:ANY,SZ:20
*      SYSRUN BIBLIO
% OPNLIB A,DVT:AD,VS:AD1MAN,LN:PROCAT,VN:0,GN:1,FT:SOU,INIT
% INCLUDE OL:*1,DL:A,OF:CATAL1
*      FETCH DVT:AD,VS:&SD:AD1MAN&,GN:1,VN:0,LN:BIBINT,FN:CREARE
*      INIT DVT:MT,VS:&SB:MT1MAN&
*      ASSIGN A,DVT:MT,VS:&SB:MT1MAN&
*      LABEL A,FN:'CONTRACTE'
*      RUN
*      FETCH DVT:AD,VS:&SD:AD1MAN&,GN:1,VN:0,LN:BIBINT,FN:CONSULT
*      ASSIGN A,DVT:MT,VS:&SB:MT1MAN&
*      LABEL A,FN:'CONTRACTE'
*      RUN
% INCLUDE OL:*1,DL:A,OF:CATAL2
*      INIT DVT:MT,VS:&SB:MT1MAN&
*      SYSRUN MAINT
&P:& &OPERATIE&
&P:%& OLDFLE DVT:&SD:AD1MAN&,FN:'BENEFICIARI
&P:%& NEWFLE DVT:&SB:MT1MAN&,FN:'BENEFICIARI',AM:ANY,SZ:5'
&P:%& END
% ENDLIB
EOJ

```

#### PROGRAMUL XXIV.1

## 2. APELAREA PROCEDURILOR CATALOGATE

Se realizează prin intermediul *cartelei* **XPROC** care are formatul următor:

$$\begin{array}{c}
 \text{XPROC} \left\{ \begin{array}{l} \text{numep} \\ * \left\{ \begin{array}{l} \text{DVT} \left\{ \begin{array}{l} \text{RD} \\ \text{MD} \\ \text{AD}' \end{array} \right\}, \text{VS:sssss} \\ \text{DV:} \left\{ \begin{array}{l} \text{RD} \\ \text{MD} \\ \text{AD} \end{array} \right\} \text{nr1} \end{array} \right\} \left\{ \begin{array}{l} [\text{MOD}] [\text{UN:nr2}] \\ [\text{GN:nr3}] [\text{VN:nr4}] \end{array} \right\} \\ [\text{FN:'numeb' SOU numep'}] \end{array} \right.
 \end{array}$$

unde:

— \* indică faptul că procedura catalogată, cu numele *numep*, se află într-o bibliotecă utilizator, cu numele *numeb*;

— MOD indică faptul că una sau mai multe *cartele* MOD urmează cartelei XPROC.

Prin intermediul cartelei MOD sînt precizate, la apelarea unei proceduri catalogate, valorile ce vor înlocui valorile parametrilor formali.

Tot prin cartela MOD, o procedură catalogată poate fi modificată (prin inserări, înlocuiri, ștergeri de linii).

Sintaxa cartelei MOD este următoarea:

$$\text{MOD} \left\{ \begin{array}{l} \text{nr1} [\text{nr2}] \\ \text{lista} \end{array} \right\}$$



unde:

— *nr1* și *nr2* au semnificația aceluiași argumente ale cartei MOD tratată de bibliotecar;

— *lista* precizează valorile efective ale parametrilor formali, astfel:

$\& p_1: v_1 \&, \dots, \& p_n: v_n \&$

unde  $p_i$  reprezintă numele parametrilor, iar  $v_i$  valorile asociate acestora.

Sfârșitul cartei MOD trebuie marcat printr-o *cartelă* ENDMOD, care nu are parametrii.

În exemplele următoare se prezintă apelarea pentru execuție a procedurilor CATAL1 și CATAL2.

```
. JOB B4,AN:ISOO,PN:FELIX
. XPROC *,DVT:AD,VS:AD1MAN,PN:'PROCAT SOU/CATAL1'
. EOJ
. JOB B5,AN:ISOO,PN:FELIX
. XPROC *,DVT:AD,VS:AD1MAN,PN:'PROCAT SOU/CATAL2',MOD
. MOD &OPERATIE:COPY,FLE&
. MOD &SB:MT2MAN& ,& SD:AD2MAN&
. ENDMOD
. EOJ
```

Se observă faptul că în cazul procedurii CATAL1 valorile inițiale ale parametrilor formali sînt păstrate pentru execuție. În cazul procedurii CATAL2 se precizează valoarea parametrului OPERATIE și se modifică valorile parametrilor SB și SD; astfel prin execuția acestei proceduri se realizează copierea fișierului BENEFICIARI de pe volumul aflat pe unitatea AD2 pe volumul aflat, pe unitatea MT2.

#### Observație

În textul unei proceduri se pot include cartele de apelare a altor proceduri. Aceste proceduri se numesc proceduri imbricate și pot fi create pe mai multe nivele. Realizarea procedurilor imbricate necesită respectarea anumitor reguli:

— definirea unui parametru formal este valabilă atîta timp cît procedura în care acesta este definit este în curs de execuție (adică definirea unui parametru în procedura X, care apelează în procedurile Y și Z) este valabilă în toate cele trei proceduri;

— la nivelul unei proceduri dacă un parametru este definit de mai multe ori, este luată în considerare ultima definire;

— la nivele diferite, definițiile externe primează celor interne.



## PARTEA ÎNTÂI

### Programarea în limbajul FORTRAN

<b>Cap. I.</b>	<b>Noțiuni introductive .....</b>	<b>3</b>
	1. Considerații generale .....	3
	2. Limbaje de programare.....	4
	3. Structura programelor FORTRAN.....	5
	4. Formularul de programare.....	7
<b>Cap. II.</b>	<b>Elemente de bază ale limbajului FORTRAN.....</b>	<b>8</b>
	1. Setul de caractere.....	8
	2. Constante .....	8
	3. Identificatori și etichete.....	10
	4. Variabile .....	10
	5. Liste de variabile.....	11
	6. Expresii .....	12
<b>Cap. III.</b>	<b>Instrucțiuni pentru scrierea programelor cu structură liniară.....</b>	<b>15</b>
	1. Instrucțiuni de declarare.....	15
	2. Instrucțiunile STOP, PAUSE, CONTINUE.....	17
	3. Instrucțiuni de atribuire.....	17
	4. Instrucțiuni de intrare/ieșire standard.....	19
<b>Cap. IV.</b>	<b>Instrucțiuni pentru scrierea programelor cu structură alternativă.....</b>	<b>29</b>
	1. Instrucțiuni de tip GO TO.....	29
	2. Instrucțiuni de tip IF.....	30
	3. Simularea structurilor IF-THEN, IF-THEN-ELSE, CASE-OF.....	31
<b>Cap. V.</b>	<b>Instrucțiuni pentru descrierea structurilor repetitive.....</b>	<b>35</b>
	1. Simularea structurii repetitive de tip DO-WHILE.....	35
	2. Simularea structurii repetitive de tip DO-UNTIL.....	36
	3. Simularea structurii repetitive condiționată intern.....	38
	4. Descrierea unei structuri repetitive de tip DO-FOR .....	39
	5. Structuri repetitive de tip DO-FOR incluse.....	41
	6. Citirea și scrierea tablourilor.....	42
<b>Cap. VI.</b>	<b>Proceduri .....</b>	<b>46</b>
	1. Funcții .....	47
	2. Subprograme .....	49
	3. Instrucțiunea EXTERNAL .....	54
<b>Cap. VII.</b>	<b>Fișiere .....</b>	<b>58</b>
	1. Utilizarea fișierelor în programele FORTRAN.....	58
	2. Declararea fișierelor .....	60



3. Fișiere pe cartele.....	62
4. Fișiere cu acces secvențial pe suport magnetic.....	63
5. Fișiere cu acces direct.....	69
Cap. VIII. Prelucrări speciale în FORTRAN.....	82
1. Instrucțiuni pentru alocarea zonelor de memorie.....	82
2. Facilități suplimentare de realizare a operațiilor de intrare/ieșire.....	86
Cap. IX. Segmentarea programelor în FORTRAN.....	91
1. Constituirea segmentelor. Tipuri de segmente. Înlănțuirea lor.....	92
2. Cartela TREE .....	93
3. Segmentarea programelor FORTRAN.....	94
Cap. X. Particularități ale limbajului FORTRAN în cazul minicalculatoarelor FELIX	102

## PARTEA A DOUA

### Programarea în limbajul COBOL

Cap. XI. Noțiuni introductive .....	106
1. Istoric .....	106
2. Caracteristici.....	107
3. Metalimbajul pentru descrierea sintaxei.....	107
Cap. XII. Elementele de bază ale limbajului COBOL.....	109
1. Structura programelor COBOL.....	109
2. Expresii și instrucțiuni aritmetice.....	126
Cap. XIII. Descrierea structurilor de control de bază în limbajul COBOL.....	136
1. Structuri secvențiale .....	136
2. Structuri alternative .....	139
3. Structuri repetitive .....	142
Cap. XIV. Elemente complementare de descriere și prelucrare a datelor.....	146
1. Reprezentarea internă a datelor. Clauza USAGE.....	146
2. Alinierea datelor. Clauza SYNCHRONIZED.....	148
3. Introducerea și extragerea de date prin instrucțiunile ACCEPT și DISPLAY .....	149
Cap. XV. Fișiere pe cartele. Validarea datelor.....	153
1. Fișiere pe cartele.....	153
2. Testarea condițiilor .....	161
3. Validarea datelor.....	166
Cap. XVI. Întocmirea rapoartelor .....	168
1. Transferul datelor .....	168
2. Editarea valorilor datelor.....	170
3. Fișiere la imprimantă.....	173
Cap. XVII. Prelucrarea tablourilor .....	178
1. Identificarea elementelor tablourilor.....	180
2. Descrierea tablourilor .....	182
3. Instrucțiuni pentru prelucrarea tablourilor.....	193
Cap. XVIII. Prelucrarea fișierelor .....	193
1. Noțiuni despre fișiere.....	195
2. Descrierea fișierelor în limbajul COBOL.....	199
3. Descrierea operațiilor de acces la fișiere și articole.....	271



4. Fișiere secvențiale pe bandă magnetică.....	202
5. Fișiere secvențiale pe disc magnetic.....	205
6. Fișiere secvențiale indexate .....	209
7. Fișiere selective .....	216
Cap. XIX. Prelucrări speciale în COBOL.....	223
1. Subprograme COBOL .....	223
2. Segmentarea programelor COBOL.....	224
3. Sortarea fișierelor în COBOL.....	228
4. Întocmirea rapoartelor prin intermediul Editorului COBOL.....	234
Cap. XX. Programul de inventariere a fișierelor și volumelor.....	245
1. Inventarierea fișierelor .....	245
2. Rezumarea volumelor.....	249
Cap. XXI. Generatorul de programe de sortare.....	252
1. Generalități .....	252
2. Ordinele transmise generatorului.....	252
Cap. XXII. Programul de întreținere a fișierelor și volumelor.....	256
1. Funcțiile programului.....	256
2. Ordinele adresate programului de întreținere.....	256
Cap. XXIII. Utilizarea bibliotecilor .....	260
1. Generalități .....	260
2. Bibliotecarul .....	260
Cap. XXIV. Utilizarea procedurilor catalogate.....	267
1. Crearea procedurilor catalogate.....	267
2. Apelarea procedurilor catalogate.....	268

Plan editură Nr. 16 705  
Coli de tipar: 17 + 11 pl.  
Bun de tipar: 8.VI.82



Tiparul executat sub comandă  
nr. 773 la  
Întreprinderea poligrafică  
„13 Decembrie 1918”,  
București  
Republica Socialistă România



ID DIVISION.  
PROGRAM-ID. PVI1.  
ENVIRONMENT DIVISION.  
CONFIGURATION SECTION.  
INPUT-OUTPUT SECTION.  
FILE-CONTROL.

SELECT MATERIALE ASSIGN TO SYSIN.  
SELECT RAPORT ASSIGN TO SYSOUT  
DATA DIVISION.  
FILE SECTION.

FD MATERIALE RECORDING F LABEL RECORD OMITTED.

01 ART-I.

05 COD-MAGAZIE PIC 99.  
05 COD-MATERIAL PIC 999.  
05 DENUMIRE-MATERIAL PIC X(20).  
05 STOC-EXISTENT PIC 9(6).  
05 PRET PIC 999V99.

FD RAPORT LABEL RECORD OMITTED.

01 ART-E PIC X(66).

WORKING-STORAGE SECTION.

01 SF PIC 9 VALUE ZERO.

01 R1.

05 FILLER PIC X(40) VALUE SPACES.  
05 FILLER PIC X(19) VALUE 'SITUATIA STOCURILOR'.  
05 FILLER PIC X(7) VALUE SPACES.

01 R2

05 FILLER PIC X(33) VALUE SPACES.  
05 FILLER PIC X(33) VALUE ALL '\*'.  
01 R3.

05 FILLER PIC X(33) VALUE SPACES.  
05 FILLER PIC X(32) VALUE '\* COD \* COD \* STOC'.  
05 FILLER PIC X VALUE '\*'.  
01 R4.

05 FILLER PIC X(33) VALUE SPACES.  
05 FILLER PIC X(32) VALUE '\* MAGAZIE \* MATERIAL \* EXISTENT'.  
05 FILLER PIC X VALUE '\*'.  
01 RIND.

05 FILLER PIC X(33) VALUE SPACES.  
05 FILLER PIC XXXX VALUE '\*'.  
05 COD-MAGAZIE PIC 99.  
05 FILLER PIC X(8) VALUE '\*'.  
05 COD-MATERIAL PIC 999.  
05 FILLER PIC X(7) VALUE '\*'.  
05 STOC-EXISTENT PIC Z(5)9.  
05 FILLER PIC XXX VALUE '\*'.  
PROCEDURE DIVISION.

PROGRAM-PRINCIPAL.

OPEN INPUT MATERIALE OUTPUT RAPORT  
READ MATERIALE AT END MOVE 1 TO SF.  
PERFORM PRELUCRARE  
CLOSE MATERIALE RAPORT  
STOP RUN.

PRELUCRARE.

IF SF = 1

THEN

DISPLAY '\*\*\* LIPSA ARTICOLE \*\*\*'

ELSE

PERFORM SCRIE-TITLU-ANTET  
PERFORM SCRIE-CITESTE UNTIL SF = 1  
PERFORM SCRIE-RIND-FINAL.

SCRIE-TITLU-ANTET.

WRITE ART-E FROM R1 AFTER 0  
WRITE ART-E FROM R2 AFTER 2  
WRITE ART-E FROM R3 AFTER 1  
WRITE ART-E FROM R4 AFTER 1  
WRITE ART-E FROM R2 AFTER 1.

SCRIE-CITESTE.

MOVE CORRESPONDING ART-E TO RIND  
WRITE ART-E FROM RIND AFTER 1  
READ MATERIALE AT END MOVE 1 TO SF

SCRIE-RIND-FINAL.

WRITE ART-E FROM R2 AFTER 1.

PROGRAMUL XVI.1



```

JOB SORTARE2,AN:100,PN:ANS
  COMPIL COBOL
  ID-DIVISION.
  PROGRAM-ID, PEX7.
  ENVIRONMENT DIVISION.
  CONFIGURATION SECTION.
  INPUT-OUTPUT SECTION.
  FILE-CONTROL.
    SELECT FISIER-CARTI ASSIGN TO SYSIN.
    SELECT BIBLIOTECA ASSIGN TO AAO
      ORGANIZATION MODE IS INDEXED
      ACCESS MODE IS SEQUENTIAL
      RECORD KEY IS COD-CARTE-B.
    SELECT MANEVRA ASSIGN TO BAD.
  DATA DIVISION.
  FILE SECTION.
  FD FISIER-CARTI RECORDING F LABEL RECORD OMITTED.
  01 ART-C.
    05 COD-CARTE-C          PIC 9(8).
    05 TITLU-C              PIC X(10).
    05 AUTOR-C              PIC A(10).
    05 NR-EXEMPLARE-C       PIC 9(5).
  FD BIBLIOTECA RECORDING F LABEL RECORD STANDARD
    BLOCK CONTAINS 20 RECORDS.
  01 ART-B.
    05 COD-CARTE-B          PIC 9(8) COMP-3.
    05 TITLU-B              PIC X(10).
    05 AUTOR-B              PIC A(10).
    05 NR-EXEMPLARE-B       PIC 9(5) COMP-3.
  SD MANEVRA DATA RECORD IS ART-M.
  01 ART-M.
    05 COD-CARTE-M          PIC 9(8) COMP-3.
    05 TITLU-M              PIC X(10).
    05 AUTOR-M              PIC A(10).
    05 NR-EXEMPLARE-M       PIC 9(5) COMP-3.
  WORKING-STORAGE SECTION.
  01 SF                      PIC 9 VALUE ZERO.
  PROCEDURE DIVISION.
  STRUCTURA SECTION.
  SORTARE.
    SORT MANEVRA
      ASCENDING KEY COD-CARTE-M
      INPUT PROCEDURE IS INTRARE
      OUTPUT PROCEDURE IS IESIRE
    STOP RUN.
  INTRARE SECTION.
  DESCHIDE-CITESTE-FISIER-CARTI.
    OPEN INPUT FISIER-CARTI
    READ FISIER-CARTI AT END MOVE 1 TO SF.
    PERFORM SCRIE UNTIL SF = 1
    CLOSE FISIER-CARTI
    GO TO INTRARE-EXIT.
  SCRIE.
    MOVE COD-CARTE-C TO COD-CARTE-M
    MOVE TITLU-C TO TITLU-M
    MOVE AUTOR-C TO AUTOR-M
    MOVE NR-EXEMPLARE-C TO NR-EXEMPLARE-M
    RELEASE ART-M
    READ FISIER-CARTI AT END MOVE 1 TO SF.
  INTRARE-EXIT.
    EXIT.
  IESIRE SECTION.
  DESCHIDE-CITESTE-MANEVRA.
    OPEN OUTPUT BIBLIOTECA
    MOVE ZERO TO SF
    RETURN MANEVRA AT END MOVE 1 TO SF.
    PERFORM CREATE-INDEXAT UNTIL NOT (SF = 0)
    CLOSE BIBLIOTECA
    GO TO IESIRE-EXIT.
  CREATE-INDEXAT.
    WRITE ART-B FROM ART-M INVALID KEY DISPLAY
      '*** CHEIE ERONATA *** ' ART
    RETURN MANEVRA AT END MOVE 1 TO SF.
  IESIRE-EXIT.
    EXIT.
  LINK
  INIT DVT:AD,VS:AD1MAN
  ALLOC DVT:AD,VS:AD1MAN,FN:'BIBLIOTECA',AN:ANY,SZ:2
  ASSIGN A,DVT:AD,VS:AD1MAN
  LABEL A,FN:'BIBLIOTECA'
  ASSIGN B,DVT:AD,VS:AD1MAN
  LABEL B,FN:'MANEVRA',AN:ANY,SZ:5
  RUN
  *
  * FISIER 'FISIER-CARTI'
  *
  EOJ

```



```

*      JOB FISIER1,AN:1100,PN:IOANA
*      COMPILE FORTRAN
C      *****
C      PROGRAMUL PRELUCREAZA UN FISIER PE
C      * CARTELE SI CREAZA UN FISIER LA
C      * IMPRIMANTA
C      *****

```

```

C      LOGICAL EOF
C      DIMENSION DEN(4),DENA(4)
C
C      SE VERIFICA DACA FISIERUL ESTE VID
C
C      READ(105,1,END=10) ICOD,DEN,CANT,PU
1  FORMAT(15,2X,4A4,2X,F5.2,2X,F4.2)
C      EOF=.FALSE.
C      GO TO 20
10  CONTINUE
C      EOF=.TRUE.
20  CONTINUE

```

```

C      IF(.NOT.EOF) GO TO 30
C      THEN
C      WRITE(108,2)
2  FORMAT(' ',10X,'FISIER VID')
C      GO TO 1000
30  CONTINUE.
C      ELSE
C
C      APELAREA SUBPROGRAMULUI CARE AFISEAZA
C      LA IMPRIMANTA CAPUL DE TABEL
C

```

```

C      CALL CAPTAB
C      ICODA=ICOD
C      TVAL=CANT*PU
C      DO 4 I=1,4
C      DENA(I)=DEN(I)
4  CONTINUE
40  CONTINUE
C      READ(105,1,END=50) ICOD,DEN,CANT,PU
C      GO TO 60
50  CONTINUE
C      EOF=.TRUE.
60  CONTINUE
C      IF(EOF) GO TO 900
C      IF(.NOT.(.ICODA.EQ.ICOD)) GO TO 70

```

```

C      THEN
C      TVAL=TVAL+CANT*PU
C      GO TO 90
70  CONTINUE
C      ELSE
C      WRITE(108,3) ICODA,DENA,TVAL
3  FORMAT(3X,' ',15,' ' ',4A4,
*      ' ',F8.2,' ')
C      ICODA=ICOD
C      DO 80 I=1,4
C      DENA(I)=DEN(I)
80  CONTINUE
C      TVAL=CANT*PU
90  CONTINUE
C      GO TO 40
900  CONTINUE
C      WRITE(108,6) ICODA,DENA,TVAL
6  FORMAT(3X,' ',15,' ' ',4A4,' ' ',
*      F8.2,' ' ',3X,' '(**))
1000 CONTINUE
C      STOP

```

```

END, SUBROUTINE CAPTAB

```

```

C      WRITE(108,1)
1  FORMAT(///3X,41(**)/3X,' ',7X,' ',18X,' ',
*      12X,' ',3X,' ',COD * DENUMIRE ' ',
*      'PRODUS * COST TOTAL */3X,' ',
*      7X,' ',18X,' ',12X,' '/3X,41(**))
C      RETURN
C      END
C      LINK
C      RUN TIME:30,NL:6000
C      100 CABLU OTEL 40001050
C      100 CABLU OTEL 15501050
C      301 CABLU SPECIAL 8001500
C      500 CABLU IZOLAT 100001725
C      .EOF
C      EOJ

```